Attorney Docket No. **09794353-0011 (formerly SUGI-T0730)** PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of:	)	Group Art Unit: **2178**
	)	
Yasuaki Yamagishi et al.	)	Examiner: **Cesar B. Paula**
	)	
Application No. **09/605,461**	)	
	)	
Filed: **June 28, 2000**	)	
	)	
For: TRANSMITTING APPARATUS, TRANSMITTING METHOD, RECEIVING APPARATUS, RECEIVING METHOD, TRANSMITTING AND RECEIVING SYSTEM, AND TRANSMITTING AND RECEIVING METHOD	)	

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANTS' BRIEF ON APPEAL

Dear Sir:

In accordance with the provisions of 37 C.F.R. § 41.37, Appellants herewith submit this Brief in support of the Appeal for the above-referenced application.

I. REAL PARTY IN INTEREST

The real party in interest in the present appeal is the Assignee, Sony Corporation, a Japanese Corporation. The Assignment was recorded in the U.S. Patent and Trademark Office at Reel 013547, Frame 0443 on December 2, 2002.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals and no related interferences.

III. STATUS OF CLAIMS

Claims 1-10 are pending in this application. The present Appeal is directed to claims 1-10 that were rejected under 35 U.S.C. § 103(a) as being unpatentable over Saether (U.S. Patent No. 6,405,219) in view of Greer (U.S. Patent No. 5,978,828) in a final office action dated November 2, 2005.

IV. STATUS OF AMENDMENTS

There are no pending amendments. However, appellants reserve the right to submit an amendment to correct noted typographical errors that do not affect the appeal.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present invention is directed to a transmitting and receiving system and method for transmitting a hierarchical structure of a directory for hierarchically managing locations of contents data and receiving the transmitted hierarchical structure. (See page 17, line 27 through page 21, line 5). The method includes the step of managing a hierarchical structure of a directory comprising a plurality of nodes, where each of the nodes is at a hierarchical level and comprises a container entry or a leaf entry. (See page 23, lines 15-24, page 37, lines 2-11; page 42, line 19 through page 43, line 6). Each of the container entries includes information of nodes at lower hierarchical levels thereof. (See page 21, lines 6-16). Each leaf entry is located

directly below one of the container entries, and does not include information of nodes at lower hierarchical levels thereof. (See page 21, lines 17-24). The method also includes the steps of detecting a change of the hierarchical structure of the directory managed by the managing step, detecting a change tracking value of the hierarchical structure on the basis of the detected change, and obtaining first difference information and second difference information. (See page 24, line 18 through page 25, line 4; page 37, lines 12-14; page 38, line 20 through page 40, line 9; page 43, lines 7-10; page 44, line 13 through page 46, line 2). The first difference information corresponds to the change of the hierarchical structure of container entries, and the second difference information corresponds to the change of the hierarchical structure of leaf entries. (See page 24, line 18 through page 25, line 4). The method further includes the step of generating first message and second message. (See page 25, lines 5-22; page 37, lines 14-17; page 40, lines 9-12; page 43, lines 11-14; page 46, lines 2-4). The first message includes the first difference information and a mask schema for interpreting a filtering mask. (See page 56, line 11 through page 63, line 9). The second message includes the second difference information and the filtering mask, where the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries. (See page 49, line 23 through page 56, line 23). The method also includes the step of adding first identification information to the first difference information and second identification information to the second difference information, and separately transmitting the first message and the second message, where the first identification information identifies each container entry, and the second identification information identifies each leaf entry. (See page 25, line 23 through page 26, line 10; page 37, lines 17-22; page 40, lines 12-14; page 43, lines 14-17; page 46, lines 4-6). The method further

includes the step of receiving the first message, the first identification information, the second message, and the second identification information transmitted in the transmitting step. (See page 29, lines 21-24; page 37, lines 23-25; page 40, lines 14-16; page 40, line 22 through page 41, line 3; page 43, lines 18-20; page 46, lines 6-9; page 46, lines 19-27). The method includes managing the hierarchical structure of the directory formed corresponding to the first message and the second message. (See page 29, line 25 through page 30, line 6; page 37, line 25 through page 38, line 5; page 40, lines 17-21; page 41, line 4 through page 42, line 18; page 43, lines 20-27; page 46, lines 10-14; page 47, line 1 through page 48, line 11). The method also includes selectively obtaining the second message and changing the hierarchical structure of the directory managed by the managing step corresponding to the obtained second message. (See page 48, line 22 through page 49, line 19).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-10 stand rejected under 35 U.S.C. § 103(a) as obvious over Saether (U.S. Patent No. 6,405,219) in view of Greer (U.S. Patent No. 5,978,828).

VII. ARGUMENT

Claims 1-10 are patentable over Saether in view of Greer.

A. The Claimed Invention

Claim 1 is directed to a transmitting apparatus for transmitting a hierarchical structure of a directory for hierarchically managing locations of contents data. The transmitting apparatus comprises managing means, detecting means, generating means, and transmitting means. The

managing means manages a hierarchical structure of a directory comprising a plurality of nodes, where each of the nodes is at a hierarchical level and comprises a container entry or a leaf entry. Each of the container entries includes information of nodes at lower hierarchical levels thereof. Each leaf entry being located directly below one of the container entries, and does not include information of nodes at lower hierarchical levels thereof. The detecting means detects a change of the hierarchical structure of the directory managed by the managing means, detects a change tracking value of the hierarchical structure on the basis of the detected change, and obtains first difference information and second difference information. The first difference information corresponds to the change of the hierarchical structure of container entries, and the second difference information corresponds to the change of the hierarchical structure of leaf entries. The generating means generates first message and second message. The first message includes the first difference information and a mask schema for interpreting a filtering mask. The second message includes the second difference information and the filtering mask, where the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries. The transmitting means separately transmits the first message and the second message.

Claims 2-3 depend from claim 1.

Claim 4 is directed to a transmitting method for transmitting a hierarchical structure of a directory for hierarchically managing locations of contents data. The transmitting method comprises the steps of: (a) managing a hierarchical structure of a directory comprising a plurality of nodes, wherein each of the nodes is at a hierarchical level and comprises a container entry or a leaf entry, each of the container entries including information of nodes at lower

hierarchical levels thereof, each leaf entry being located directly below one of the container entries, each leaf entry not including information of nodes at lower hierarchical levels thereof; (b) detecting a change of the hierarchical structure of the directory managed at step (a), detecting a change tracking value of the hierarchical structure on the basis of the detected change, and obtaining first difference information and second difference information, the first difference information corresponding to the change of the hierarchical structure of container entries, the second difference information corresponding to the change of the hierarchical structure of leaf entries; (c) generating first message and second message, the first message including the first difference information and a mask schema for interpreting a filtering mask, the second message including the second difference information and the filtering mask, wherein the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries; and (d) separately transmitting the first message and the second message.

Claim 5 is directed to a receiving apparatus for receiving a hierarchical structure of a directory for hierarchically managing locations of contents data that is transmitted. The receiving apparatus comprises receiving means, managing means and changing means. The receiving means receives first message, first identification information, second message, and second identification information. The first message includes first difference information and the second message includes second difference information. The first difference information is obtained by detecting a change of container entries. The first identification information identifies each container entry added to the first difference information. The second difference information is obtained by detecting a change of leaf entries. The second identification information identifies each leaf entry added to the second difference information. The directory

comprises a plurality of nodes, where each of the nodes is at a hierarchy level and comprises one of the container entries or one of the leaf entries. Each of the container entries includes information of nodes at lower hierarchical levels thereof. Each leaf entry is located directly below one of the container entries, and does not include information of nodes at lower hierarchical levels thereof. The first message further includes a mask schema for interpreting a filtering mask. The second message further including the filtering mask, where the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries. The managing means manages the hierarchical structure of the directory formed corresponding to the first message and the second message. The changing means selectively obtains the second message and changes the hierarchical structure of the directory managed by the managing means corresponding to the obtained second message.

Claim 6 depends from claim 5.

Claim 7 is directed to a receiving method for receiving a hierarchical structure of a directory for hierarchically managing locations of contents data that is transmitted. The receiving method comprises the steps of: (a) receiving first message, first identification information, second message, and second identification information, wherein the first message includes first difference information and the second message includes second difference information, the first difference information being obtained by detecting a change of container entries, the first identification information identifying each container entry added to the first difference information, the second difference information being obtained by detecting a change of leaf entries, the second identification information identifying each leaf entry added to the second difference information, the directory comprising a plurality of nodes, wherein each of the

nodes is at a hierarchy level and comprises one of the container entries or one of the leaf entries, each of the container entries including information of nodes at lower hierarchical levels thereof, each leaf entry being located directly below one of the container entries, each leaf entry not including information of nodes at lower hierarchical levels thereof, the receiving means, the first message further including a mask schema for interpreting a filtering mask, the second message further including the filtering mask, wherein the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries; (b) managing the hierarchical structure of the directory formed corresponding to the first message and the second message; and (c) selectively obtaining the second message and changing the hierarchical structure of the directory managed at step (b) corresponding to the obtained second message.

Claim 8 is directed to a transmitting and receiving system for transmitting a hierarchical structure of a directory for hierarchically managing locations of contents data and receiving the transmitted hierarchical structure. The transmitting and receiving system comprises first managing means, detecting means, generating means, transmitting means, receiving means, second managing means and changing means. The first managing means manages a hierarchical structure of a directory comprising a plurality of nodes, where each of the nodes is at a hierarchical level and comprises a container entry or a leaf entry. Each of the container entries includes information of nodes at lower hierarchical levels thereof. Each leaf entry is located directly below one of the container entries, and does not include information of nodes at lower hierarchical levels thereof. The detecting means detects a change of the hierarchical structure of the directory managed by the managing means, detects a change tracking value of the hierarchical structure on the basis of the detected change, and obtains first difference information

and second difference information. The first difference information corresponds to the change of the hierarchical structure of container entries, and the second difference information corresponds to the change of the hierarchical structure of leaf entries. The generating means generates first message and second message. The first message includes the first difference information and a mask schema for interpreting a filtering mask. The second message includes the second difference information and the filtering mask, where the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries. The transmitting means adds first identification information to the first difference information and second identification information to the second difference information, and separately transmits the first message and the second message. The first identification information identifies each container entry, and the second identification information identifies each leaf entry. The receiving means receives the first message, the first identification information, the second message, and the second identification information transmitted by the transmitting means. The second managing means manages the hierarchical structure of the directory formed corresponding to the first message and the second message. The changing means selectively obtains the second message and changes the hierarchical structure of the directory managed by the second managing means corresponding to the obtained second message.

Claim 9 depends from claim 8.

Claim 10 is directed to a transmitting and receiving method for transmitting and receiving a hierarchical structure of a directory for hierarchically managing locations of contents data and receiving the transmitted hierarchical structure. The transmitting and receiving method comprises the steps of: (a) managing a hierarchical structure of a directory comprising a

plurality of nodes, wherein each of the nodes is at a hierarchical level and comprises a container entry or a leaf entry, each of the container entries including information of nodes at lower hierarchical levels thereof, each leaf entry being located directly below one of the container entries, each leaf entry not including information of nodes at lower hierarchical levels thereof; (b) detecting a change of the hierarchical structure of the directory managed at step (a), detecting a change tracking value of the hierarchical structure on the basis of the detected change, and obtaining first difference information and second difference information, the first difference information corresponding to the change of the hierarchical structure of container entries, the second difference information corresponding to the change of the hierarchical structures of leaf entries; (c) generating first message and second message, the first message including the first difference information and a mask schema for interpreting a filtering mask, the second message including the second difference information and the filtering mask wherein the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries; (d) adding first identification information to the first difference information and second identification information to the second difference information and separately transmitting the first message and the second message, the first identification information identifying each container entry, the second identification information identifying each leaf entry; (e) receiving the first message, the first identification information, the second message, and the second identification information transmitted at step (c); (f) managing the hierarchical structure of the directory formed corresponding to the first message and the second message; and (g) selectively obtaining the second message and changing the hierarchical structure of the directory managed at step (f) corresponding to the obtained second message.

B. Claims 1-10 Are Patentable

In the Final Office Action, claims 1-10 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Saether (U.S. Patent No. 6,405,219) in view of Greer (U.S. Patent No. 5,978,828). The Examiner has not made an adequate showing to support his rejections.

Saether is directed to a method and system for automatically updating the version of a set of files stored on content servers. Saether provides for managing the distribution and synchronization of a set of source files for remotely located content servers according to the directory structure and hardware configuration of each content server. (See col. 4, lines 28-34.) According to the Examiner, the files (A, B, & C) in Saether correspond to the leaf entries, and the directories (root and subdirectory D1) correspond to the container entries. (See col. 10, lines 39-45 and Fig. 5A.) Thus, Saether discloses transmitting and receiving changes to a hierarchical structure, but does not disclose or suggest the details required by the claims.

Greer is directed to an apparatus and method of providing update notification of Web page content or location changes. (See col. 2, lines 25-27.) In Greer, a Web page 200 includes objects such as HTML links, motion images, sound clips, HTML page tables, and other embedded objects. (See col. 3, lines 24-33 and Fig. 3.) Greer discloses a change control record 300, which includes a URL field 304 and a global quotient field 306. (See col. 3, lines 40-47 and Fig. 4.) The URL field 304 specifies whether the address of the Web page has changed. (See col. 3, lines 49-51.) The global quotient field 306 includes a global quotient value that specifies the magnitude of change of the overall Web page since the last update. (See col. 3, lines 51-54.) The Web page change control record 300 further includes one or more object fields 320 corresponding to the number of Web page objects. (See col. 4, lines 7-9.) Each object field 320

includes an object quotient field 328, which includes an object quotient value that specifies the magnitude of change of the particular object since the last update. (See col. 4, lines 13-25.) Greer discloses that the quotient page 500 includes an optional URL field 502, which includes a new address of the Web page when the URL of the Web page has changed. (See col. 6, lines 14-16.)

The Examiner claims that: (1) the web page corresponds to the container entry and the different objects of the web page correspond to the leaf entries; (2) the date and time of the last web page modification correspond to the first difference information; (3) the object quotient field followed by the date and time correspond to the second difference information; (4) the URL field containing a quotient page with a URL corresponds to the filtering mask; and (5) the global quotient value corresponds to the mask schema. Contrary to the Examiner's statement, the URL field does not contain the quotient page. Rather, the quotient page includes the URL field. (See col. 6, lines 14-16.)

As discussed above, the URL field in Greer specifies whether the address of the web page has changed. (See col. 3, lines 49-51). Greer does not disclose or suggest whether the URL field informs the user when an object within the web page has been changed or updated. Accordingly, the URL field in Greer does not correspond to information of one of the leaf entries (*i.e.*, objects) being directly under one of the container entries (*i.e.*, web pages), as required by the claims. Moreover, because the global quotient value in Greer does not interpret the URL field, Greer does not disclose or suggest a mask schema for interpreting a filter mask, as required by the claims.

Finally, although the Examiner claims that the global quotient value in Greer corresponds to the "mask schema," in the 2/24/06 Advisory Action, the Examiner states that when the new web page address (which the Examiner claims is the "filtering mask") is not found, the quotient page, along with the global quotient value and date and time of last modification (which the Examiner claims are the "first message") is transmitted, and when the "filtering mask" is found, the "filtering mask" is transmitted with the global quotient value (*i.e.*, the "mask schema") and object quotient value (*i.e.*, the "second difference information"). Thus, the Examiner seems to indicate that when the "filtering mask" is found, only the "first message" is transmitted, and if the "filtering mask" is not found, the "mask schema," which is part of the "first message," is transmitted with the "second message." Thus, the Examiner does not indicate how the first message and second message are transmitted separately in Greer.

For the reasons set forth above, Greer does not disclose or suggest generating a first message including the first difference information and a mask schema for interpreting a filtering mask, the second message including the second difference information and the filtering mask, wherein the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries or transmitting means for separately transmitting the first message and the second message, as required by the claims, and it would not have been obvious to one of ordinary skill in the art at the time of the invention to have combined the teachings of Saether and Greer to derive claims 1-10. Accordingly, Applicants respectfully submit that claims 1-10 are allowable over Saether in view of Greer.

In view of the foregoing, Appellants respectfully submit that claims 1-10 are patentable and the application is in condition for allowance.

C. Conclusion

Appellants respectfully submit that the subject matter of the claims on appeal is not disclosed or suggested by Saether or Greer. Thus, the Examiner has not made an adequate showing of obviousness with respect to the subject matter of the rejected claims. Appellants, therefore, respectfully request reversal of the Examiner's decision to reject claims 1-10 under 35 U.S.C. § 103(a) as being unpatentable over Saether in view of Greer, and respectfully request allowance of all pending claims.

Respectfully submitted,

Dated: May 31, 2006

By: _____

Marina N. Saito
Registration No. 42,121
SONNENSCHEIN NATH & ROSENTHAL LLP
P.O. Box 061080
Wacker Drive Station, Sears Tower
Chicago, Illinois 60606-1080
(312) 876-8000

VIII. CLAIMS APPENDIX

1. (Previously Presented) A transmitting apparatus for transmitting a hierarchical structure of a directory for hierarchically managing locations of contents data, comprising:

managing means for managing a hierarchical structure of a directory comprising a plurality of nodes, wherein each of said nodes is at a hierarchical level and comprises a container entry or a leaf entry, each of said container entries including information of nodes at lower hierarchical levels thereof, each leaf entry being located directly below one of the container entries, each leaf entry not including information of nodes at lower hierarchical levels thereof;

detecting means for detecting a change of the hierarchical structure of the directory managed by said managing means, detecting a change tracking value of said hierarchical structure on the basis of the detected change, and obtaining first difference information and second difference information, the first difference information corresponding to the change of the hierarchical structure of container entries, the second difference information corresponding to the change of the hierarchical structure of leaf entries;

generating means for generating first message and second message, the first message including said first difference information and a mask schema for interpreting a filtering mask, the second message including said second difference information and the filtering mask, wherein the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries; and

transmitting means for separately transmitting said first message and said second message.

2. (Previously Presented) The transmitting apparatus as set forth in claim 1,

wherein said filtering mask comprises a value based on number of container entries on the hierarchical level directly above the one leaf entry.

3. (Original) The transmitting apparatus as set forth in claim 1,

wherein said transmitting means adds identification information that identifies each leaf entry to the second difference information.

4. (Previously Presented) A transmitting method for transmitting a hierarchical structure of a directory for hierarchically managing locations of contents data, comprising the steps of:

(a) managing a hierarchical structure of a directory comprising a plurality of nodes, wherein each of said nodes is at a hierarchical level and comprises a container entry or a leaf entry, each of said container entries including information of nodes at lower hierarchical levels thereof, each leaf entry being located directly below one of the container entries, each leaf entry not including information of nodes at lower hierarchical levels thereof;

(b) detecting a change of the hierarchical structure of the directory managed at step (a), detecting a change tracking value of said hierarchical structure on the basis of the detected change, and obtaining first difference information and second difference information, the first difference information corresponding to the change of the hierarchical structure of container entries, the second difference information corresponding to the change of the hierarchical structure of leaf entries;

(c) generating first message and second message, the first message including said first difference information and a mask schema for interpreting a filtering mask, the second message

including said second difference information and the filtering mask, wherein the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries; and

(d) separately transmitting said first message and said second message.

5. (Previously Presented) A receiving apparatus for receiving a hierarchical structure of a directory for hierarchically managing locations of contents data that is transmitted, comprising:

receiving means for receiving first message, first identification information, second message, and second identification information, wherein said first message includes first difference information and said second message includes second difference information, the first difference information being obtained by detecting a change of container entries, the first identification information identifying each container entry added to the first difference information, the second difference information being obtained by detecting a change of leaf entries, the second identification information identifying each leaf entry added to the second difference information, the directory comprising a plurality of nodes, wherein each of said nodes is at a hierarchy level and comprises one of the container entries or one of the leaf entries, each of said container entries including information of nodes at lower hierarchical levels thereof, each leaf entry being located directly below one of the container entries, each leaf entry not including information of nodes at lower hierarchical levels thereof, said receiving means, the first message further including a mask schema for interpreting a filtering mask, the second message further including the filtering, wherein the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries;

managing means for managing the hierarchical structure of the directory formed corresponding to the first message and the second message; and

changing means for selectively obtaining the second message and changing the hierarchical structure of the directory managed by said managing means corresponding to the obtained second message.

6. (Previously Presented) The receiving apparatus as set forth in claim 5,

wherein said changing means selectively obtains the second message as one of the leaf entries of the lower hierarchical level of a container entry represented by the first identification information corresponding to selection information selectively designated to the first identification information and changes the hierarchical structure of the directory managed by said managing means.

7. (Previously Presented) A receiving method for receiving a hierarchical structure of a directory for hierarchically managing locations of contents data that is transmitted, comprising the steps of:

(a) receiving first message, first identification information, second message, and second identification information, wherein said first message includes first difference information and said second message includes second difference information, the first difference information being obtained by detecting a change of container entries, the first identification information identifying each container entry added to the first difference information, the second difference information being obtained by detecting a change of leaf entries, the second identification information identifying each leaf entry added to the second difference information, the directory comprising a plurality of nodes, wherein each of said nodes is at a hierarchy level and comprises

one of the container entries or one of the leaf entries, each of said container entries including information of nodes at lower hierarchical levels thereof, each leaf entry being located directly below one of the container entries, each leaf entry not including information of nodes at lower hierarchical levels thereof, said receiving means, the first message further including a mask schema for interpreting a filtering mask, the second message further including the filtering mask, wherein the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries;

(b) managing the hierarchical structure of the directory formed corresponding to the first message and the second message; and

(c) selectively obtaining the second message and changing the hierarchical structure of the directory managed at step (b) corresponding to the obtained second message.

8. (Previously Presented) A transmitting and receiving system for transmitting a hierarchical structure of a directory for hierarchically managing locations of contents data and receiving the transmitted hierarchical structure, comprising:

first managing means for managing a hierarchical structure of a directory comprising a plurality of nodes, wherein each of said nodes is at a hierarchical level and comprises a container entry or a leaf entry, each of said container entries including information of nodes at lower hierarchical levels thereof, each leaf entry being located directly below one of said container entries, each leaf entry not including information of nodes at lower hierarchical levels thereof;

detecting means for detecting a change of the hierarchical structure of the directory managed by said managing means, detecting a change tracking value of said hierarchical structure on the basis of the detected change, and obtaining first difference information and

second difference information, the first difference information corresponding to the change of the hierarchical structure of container entries, the second difference information corresponding to the change of the hierarchical structure of leaf entries;

generating means for generating first message and second message, the first message including said first difference information and a mask schema for interpreting a filtering mask, the second message including said second difference information and the filtering mask wherein the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries;

transmitting means for adding first identification information to the first difference information and second identification information to the second difference information and separately transmitting said first message and said second message, the first identification information identifying each container entry, the second identification information identifying each leaf entry;

receiving means for receiving the first message, the first identification information, the second message, and the second identification information transmitted by said transmitting means;

second managing means for managing the hierarchical structure of the directory formed corresponding to the first message and the second message; and

changing means for selectively obtaining the second message and changing the hierarchical structure of the directory managed by said second managing means corresponding to the obtained second message.

9. (Previously Presented) The receiving and transmitting system as set forth in claim 8,

wherein said changing means selectively obtains the second difference information as one of the leaf entries of the lower hierarchical level of a container entry represented by the first identification information corresponding to selection information selectively designated to the first identification information and changes the hierarchical structure of the directory managed by said second managing means.

10. (Previously Presented) A transmitting and receiving method for transmitting a hierarchical structure of a directory for hierarchically managing locations of contents data and receiving the transmitted hierarchical structure, comprising the steps of:

(a) managing a hierarchical structure of a directory comprising a plurality of nodes, wherein each of said nodes is at a hierarchical level and comprises a container entry or a leaf entry, each of said container entries including information of nodes at lower hierarchical levels thereof, each leaf entry being located directly below one of said container entries, each leaf entry not including information of nodes at lower hierarchical levels thereof;

(b) detecting a change of the hierarchical structure of the directory managed at step (a), detecting a change tracking value of said hierarchical structure on the basis of the detected change, and obtaining first difference information and second difference information, the first difference information corresponding to the change of the hierarchical structure of container entries, the second difference information corresponding to the change of the hierarchical structures of leaf entries;

(c) generating first message and second message, the first message including said first difference information and a mask schema for interpreting a filtering mask, the second message including said second difference information and the filtering mask wherein the filtering mask corresponds to information of one of the leaf entries being directly under one of the container entries;

(d) adding first identification information to the first difference information and second identification information to the second difference information and separately transmitting said first message and said second message, the first identification information identifying each container entry, the second identification information identifying each leaf entry;

(e) receiving the first message, the first identification information, the second message, and the second identification information transmitted at step (c);

(f) managing the hierarchical structure of the directory formed corresponding to the first message and the second message; and

(g) selectively obtaining the second message and changing the hierarchical structure of the directory managed at step (f) corresponding to the obtained second message.

X. EVIDENCE APPENDIX

Appellants attach hereto copies of the patents to (1) Saether (U.S. Patent No. 6,405,219), and (2) Greer (U.S. Patent No. 5,978,828), which were relied upon by the Examiner in his rejection entered on November 2, 2005.

US006405219B2

(12) **United States Patent**
Saether et al.

(10) **Patent No.: US 6,405,219 B2**
(45) **Date of Patent: *Jun. 11, 2002**

(54) **METHOD AND SYSTEM FOR AUTOMATICALLY UPDATING THE VERSION OF A SET OF FILES STORED ON CONTENT SERVERS**

(75) Inventors: **Christian D. Saether**; **David E. Sloat**, both of Seattle, WA (US)

(73) Assignee: **F5 Networks, Inc.**, Seattle, WA (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/405,894**

(22) Filed: **Sep. 24, 1999**

Related U.S. Application Data

(60) Provisional application No. 60/140,213, filed on Jun. 22, 1999.

(51) **Int. Cl.**[7] **G06F 17/30**

(52) **U.S. Cl.** **707/201**; 709/219; 709/232; 709/246

(58) **Field of Search** 707/10, 101, 202–205; 709/219

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,950,735	A	4/1976	Patel	340/172.5
4,644,532	A	2/1987	George et al.	370/94
4,965,772	A	10/1990	Daniel et al.	364/900

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	0 744 850 A2	11/1996
WO	WO 91/14326	9/1991
WO	WO 95/05712	2/1995

(List continued on next page.)

OTHER PUBLICATIONS

"Servlet/Applet/HTML Authentication Process With Single Sign–On," *Research Disclosure 429128*, IBM Corporation, pp. 163–164, Jan. 2000.

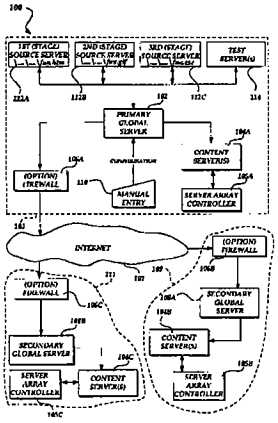
"A Process for Selective Routing of Serviet Content to Transcoding Modules," *Research Disclosure 422124*, IBM Corporation, pp. 889–890, Jun., 1999.

Primary Examiner—Safet Metjahic
Assistant Examiner—Haythim Alaubaidi
(74) *Attorney, Agent, or Firm*—Merchant & Gould P.C.; John W. Branch

(57) **ABSTRACT**

A method and system for managing the replication and version synchronization of updates to a set of source files on geographically distributed heterogeneous content servers with minimal impact on a network's bandwidth. The configuration of each content server is either manually entered or automatically determined. The current version of the source files are created on at least one source server. A Primary global server stores a copy of the current version of the set of the source files along with the configuration of each content server. The Primary global server generates and distributes a particular version change container and version distribution list to each remotely located Secondary global server. Each Secondary global server employs the version distribution list and the contents of the version change container to identify the current version of each source file necessary to upgrade the set of source files on each local content server. Each identified source file is copied to a sub-directory on each local content server associated with the Secondary global server. At each local content server, the renaming of each copied source file is employed to update to the current version of the set of source files on the content server. A versioned file tree repository for the set of source files includes archived objects. When the version distribution list identifies a previous version, the current version of source files on the local content servers can be rolled back to the previous version.

42 Claims, 9 Drawing Sheets



U.S. PATENT DOCUMENTS

5,023,826	A		6/1991	Patel 364/736
5,053,953	A		10/1991	Patel 364/200
5,299,312	A		3/1994	Rocco, Jr. 395/200
5,327,529	A		7/1994	Fults et al. 395/155
5,367,635	A		11/1994	Bauer et al.
5,371,852	A		12/1994	Attanasio et al. 395/200
5,406,502	A		4/1995	Haramaty 364/551.1
5,475,857	A		12/1995	Dally 395/800
5,517,617	A		5/1996	Sathaye et al. 395/200.1
5,519,694	A		5/1996	Brewer et al. 370/54
5,519,778	A		5/1996	Leighton et al. 380/30
5,521,591	A		5/1996	Arora et al. 340/826
5,528,701	A		6/1996	Aref 382/178
5,581,764	A		12/1996	Fitzgerald et al.
5,596,742	A		1/1997	Agarwal et al. 395/500
5,606,665	A		2/1997	Yang et al. 395/200.2
5,611,049	A		3/1997	Pitts 395/200.9
5,663,018	A		9/1997	Cummings et al. 430/5
5,678,042	A	*	10/1997	Pisello et al. 707/10
5,752,023	A		5/1998	Chourci et al. 395/610
5,761,484	A		6/1998	Agarwal et al. 395/500
5,768,423	A		6/1998	Aref et al. 384/228
5,774,660	A		6/1998	Brendel et al. 395/200.31
5,778,395	A	*	7/1998	Whiting et al. 707/204
5,790,554	A		8/1998	Pitcher et al. 3704/471
5,875,296	A		2/1999	Shi et al. 395/188.01
5,892,914	A		4/1999	Pitts 395/200.49
5,919,247	A		7/1999	Van Hoff et al.
5,933,834	A	*	8/1999	Aichelen 707/103
5,936,939	A		8/1999	Des Jardins et al. 370/229
5,946,690	A		8/1999	Pitts 707/10
5,949,885	A		9/1999	Leighton 380/54
5,959,990	A		9/1999	Frantz et al. 370/392
5,974,460	A		10/1999	Maddalozzo, Jr. et al. . 709/224
5,983,281	A		11/1999	Ogle et al. 709/249
6,006,260	A		12/1999	Barrick, Jr. et al. 709/224
6,006,264	A		12/1999	Colby et al. 709/226
6,026,452	A		2/2000	Pitts 710/56
6,028,857	A		2/2000	Poor 370/351
6,051,169	A		4/2000	Brown et al. 264/40.1
6,076,105	A	*	6/2000	Wolff et al. 707/10
6,078,956	A		6/2000	Bryant et al. 709/224
6,085,234	A		7/2000	Pitts 709/217
6,092,196	A		7/2000	Reiche 713/200
6,108,703	A		8/2000	Leighton et al. 709/226
6,111,876	A		8/2000	Frantz et al. 370/392
6,145,011	A	*	11/2000	Furukawa et al. 709/245
6,173,293	B1	*	1/2001	Thekkath et al. 707/201
6,298,319	B1	*	10/2001	Heile et al. 703/26

FOREIGN PATENT DOCUMENTS

WO	WO 97/09805	3/1997
WO	WO 97/45800	12/1997
WO	WO 99/05829	2/1999
WO	WO 99/06913	2/1999
WO	WO 99/10858	3/1999
WO	WO 99/39373	8/1999
WO	WO 99/64967	12/1999
WO	WO 00/04422	1/2000
WO	WO 00/04458	1/2000

* cited by examiner

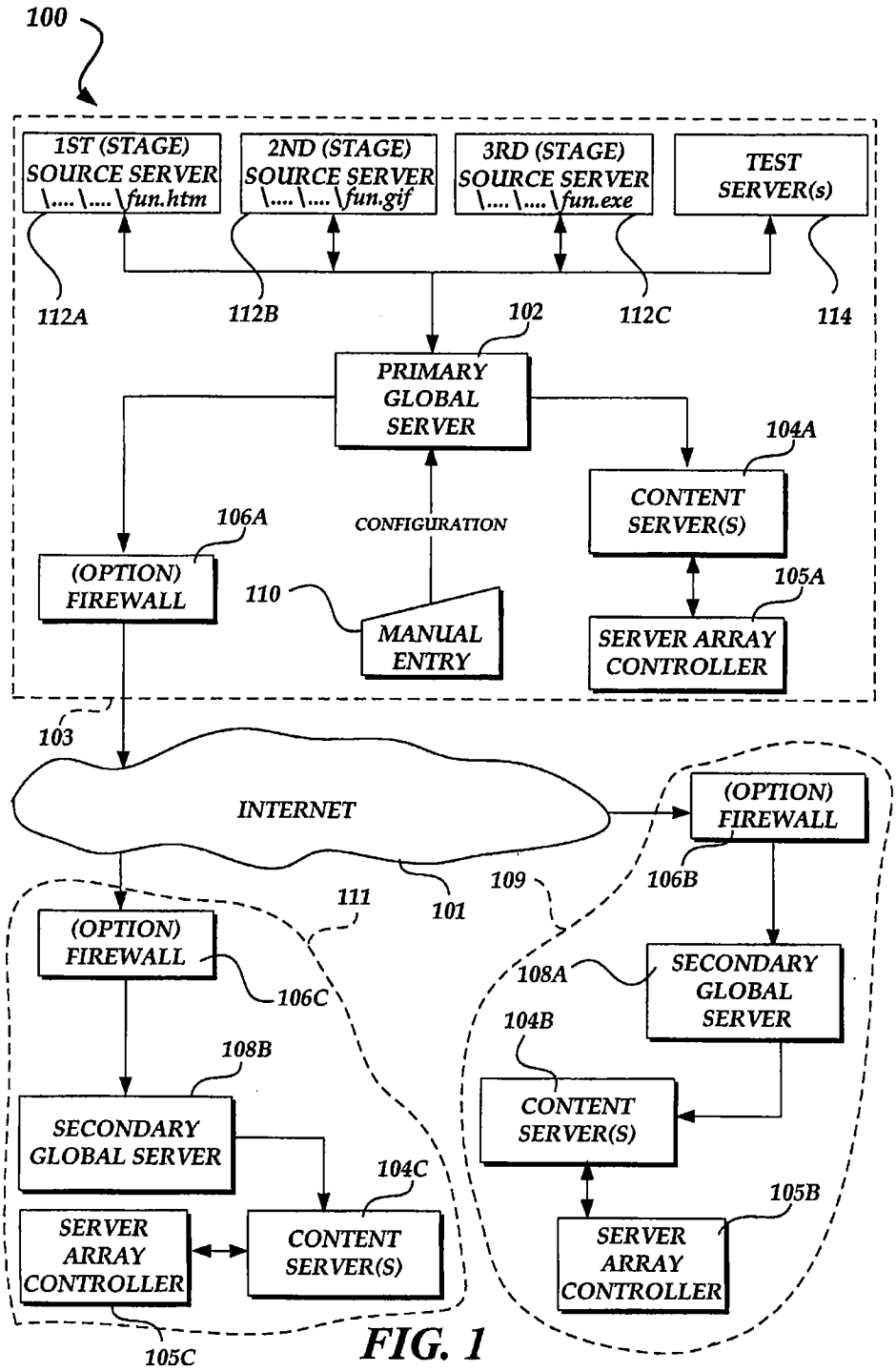

100
1ST (STAGE) SOURCE SERVER \....\....\fun.htm
2ND (STAGE) SOURCE SERVER \....\....\fun.gif
3RD (STAGE) SOURCE SERVER \....\....\fun.exe
TEST SERVER(s)
112A
112B
112C
114
102
PRIMARY GLOBAL SERVER
104A
CONTENT SERVER(S)
106A
(OPTION) FIREWALL
CONFIGURATION
110
MANUAL ENTRY
105A
SERVER ARRAY CONTROLLER
103
INTERNET
(OPTION) FIREWALL
106B
109
101
111
(OPTION) FIREWALL
106C
108A
SECONDARY GLOBAL SERVER
104B
CONTENT SERVER(S)
108B
SECONDARY GLOBAL SERVER
104C
SERVER ARRAY CONTROLLER
CONTENT SERVER(S)
105B
SERVER ARRAY CONTROLLER
105C

FIG. 1

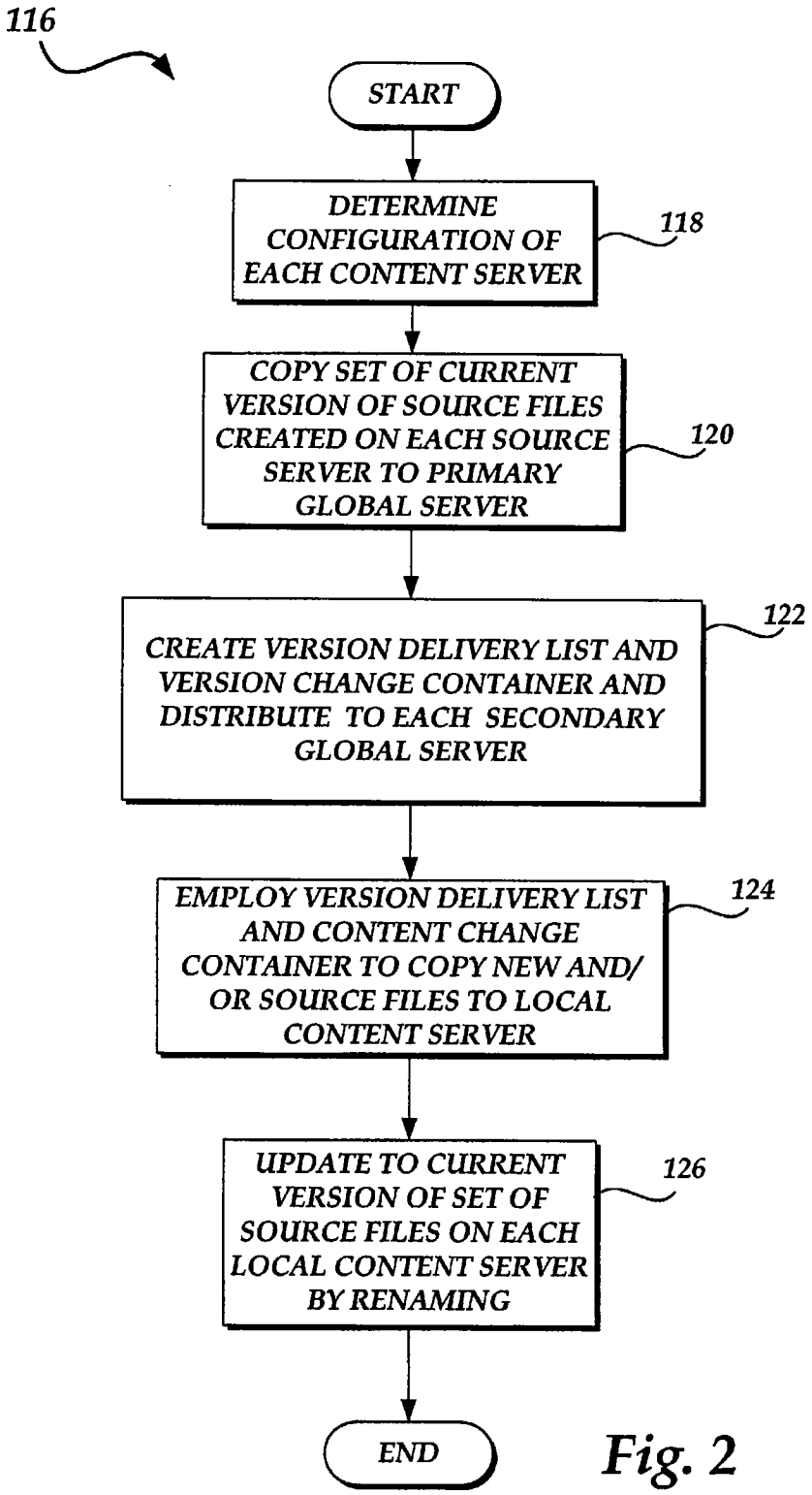

116
START
DETERMINE CONFIGURATION OF EACH CONTENT SERVER
118
COPY SET OF CURRENT VERSION OF SOURCE FILES CREATED ON EACH SOURCE SERVER TO PRIMARY GLOBAL SERVER
120
CREATE VERSION DELIVERY LIST AND VERSION CHANGE CONTAINER AND DISTRIBUTE TO EACH SECONDARY GLOBAL SERVER
122
EMPLOY VERSION DELIVERY LIST AND CONTENT CHANGE CONTAINER TO COPY NEW AND/ OR SOURCE FILES TO LOCAL CONTENT SERVER
124
UPDATE TO CURRENT VERSION OF SET OF SOURCE FILES ON EACH LOCAL CONTENT SERVER BY RENAMING
126
END

Fig. 2

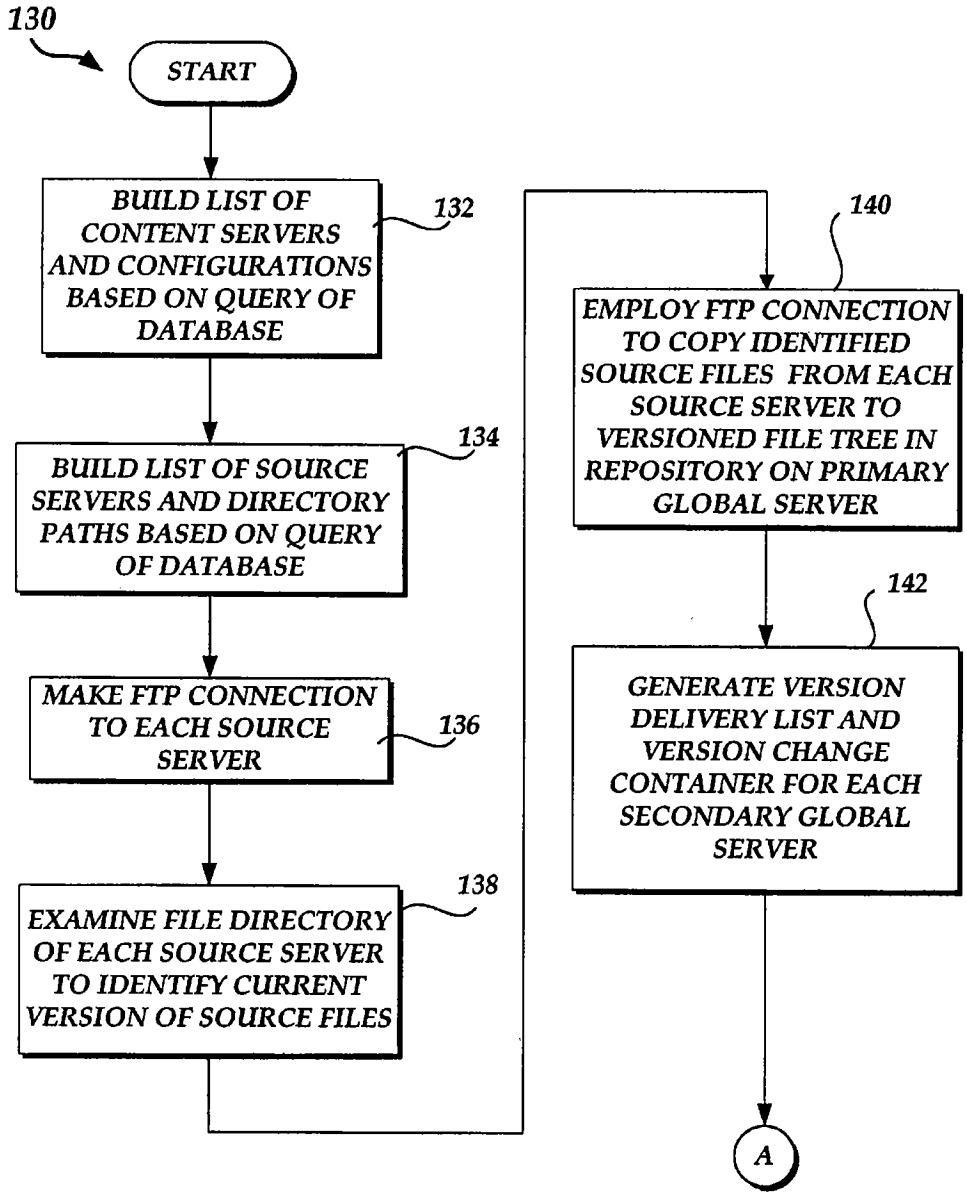

130
START
BUILD LIST OF CONTENT SERVERS AND CONFIGURATIONS BASED ON QUERY OF DATABASE
132
BUILD LIST OF SOURCE SERVERS AND DIRECTORY PATHS BASED ON QUERY OF DATABASE
134
MAKE FTP CONNECTION TO EACH SOURCE SERVER
136
EXAMINE FILE DIRECTORY OF EACH SOURCE SERVER TO IDENTIFY CURRENT VERSION OF SOURCE FILES
138
EMPLOY FTP CONNECTION TO COPY IDENTIFIED SOURCE FILES FROM EACH SOURCE SERVER TO VERSIONED FILE TREE IN REPOSITORY ON PRIMARY GLOBAL SERVER
140
GENERATE VERSION DELIVERY LIST AND VERSION CHANGE CONTAINER FOR EACH SECONDARY GLOBAL SERVER
142
A

Fig. 3A

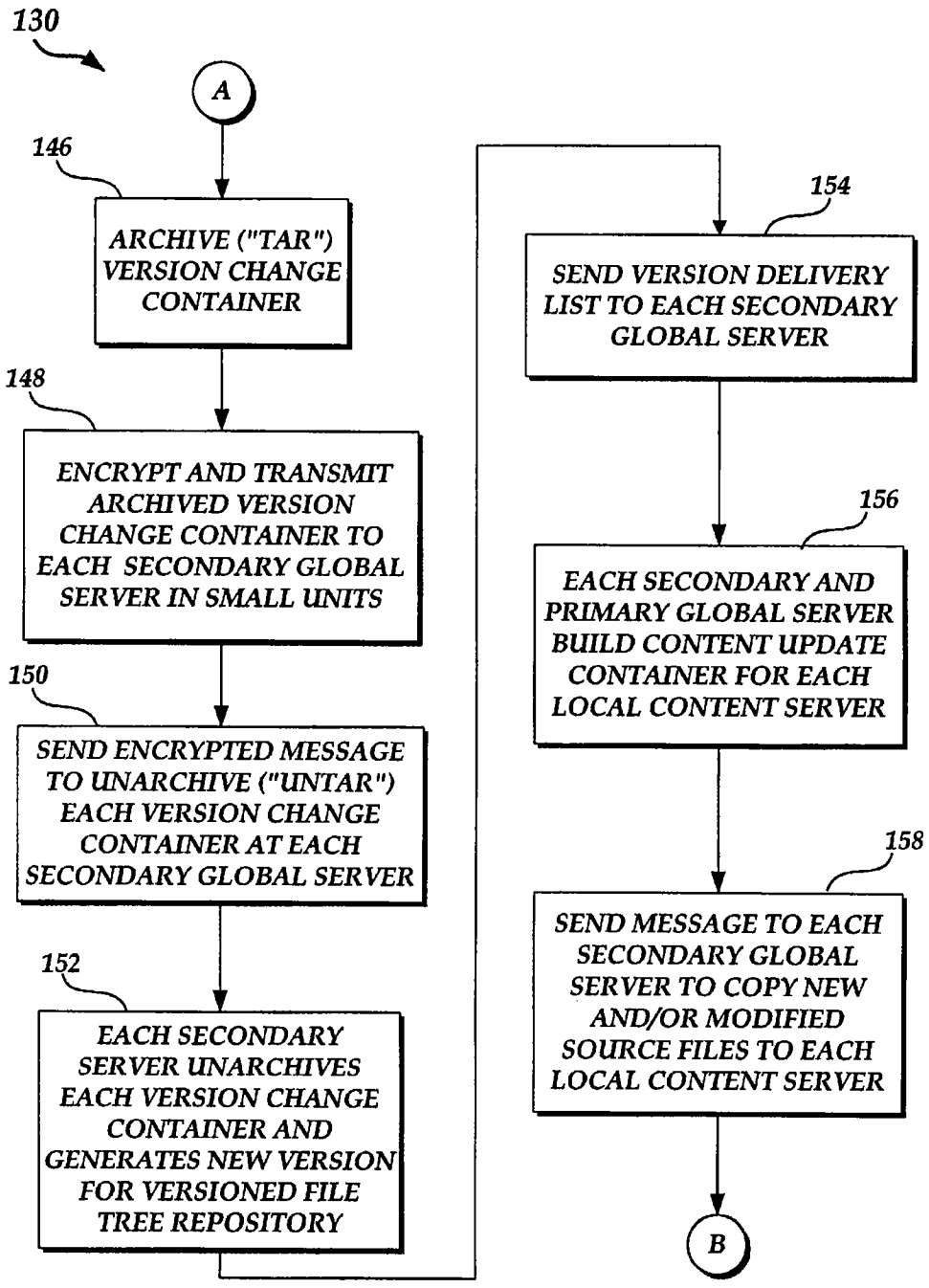

130
A
146
ARCHIVE ("TAR") VERSION CHANGE CONTAINER
148
ENCRYPT AND TRANSMIT ARCHIVED VERSION CHANGE CONTAINER TO EACH SECONDARY GLOBAL SERVER IN SMALL UNITS
150
SEND ENCRYPTED MESSAGE TO UNARCHIVE ("UNTAR") EACH VERSION CHANGE CONTAINER AT EACH SECONDARY GLOBAL SERVER
152
EACH SECONDARY SERVER UNARCHIVES EACH VERSION CHANGE CONTAINER AND GENERATES NEW VERSION FOR VERSIONED FILE TREE REPOSITORY
154
SEND VERSION DELIVERY LIST TO EACH SECONDARY GLOBAL SERVER
156
EACH SECONDARY AND PRIMARY GLOBAL SERVER BUILD CONTENT UPDATE CONTAINER FOR EACH LOCAL CONTENT SERVER
158
SEND MESSAGE TO EACH SECONDARY GLOBAL SERVER TO COPY NEW AND/OR MODIFIED SOURCE FILES TO EACH LOCAL CONTENT SERVER
B

Fig. 3B

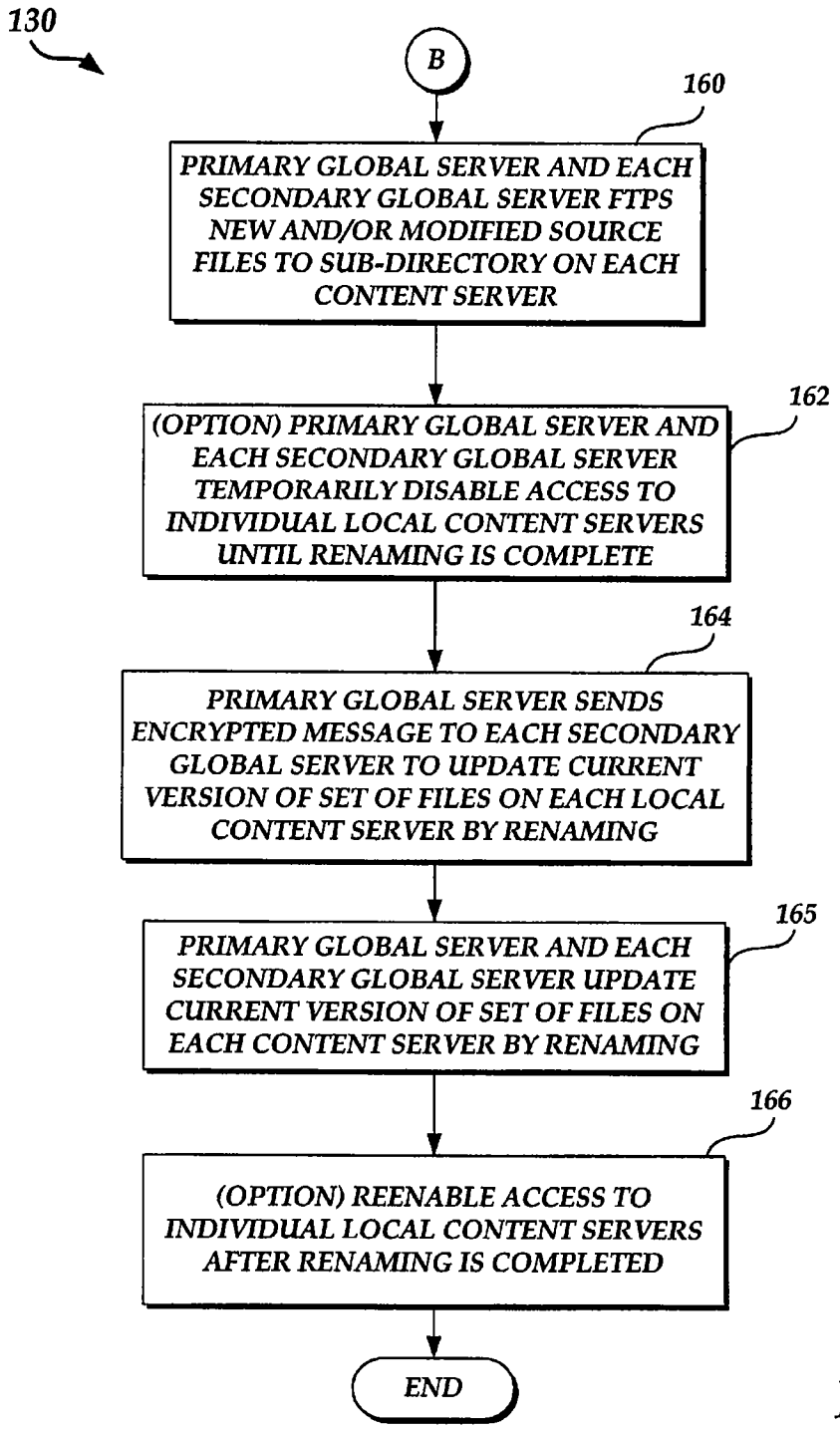

130
B
160
PRIMARY GLOBAL SERVER AND EACH SECONDARY GLOBAL SERVER FTPS NEW AND/OR MODIFIED SOURCE FILES TO SUB-DIRECTORY ON EACH CONTENT SERVER
162
(OPTION) PRIMARY GLOBAL SERVER AND EACH SECONDARY GLOBAL SERVER TEMPORARILY DISABLE ACCESS TO INDIVIDUAL LOCAL CONTENT SERVERS UNTIL RENAMING IS COMPLETE
164
PRIMARY GLOBAL SERVER SENDS ENCRYPTED MESSAGE TO EACH SECONDARY GLOBAL SERVER TO UPDATE CURRENT VERSION OF SET OF FILES ON EACH LOCAL CONTENT SERVER BY RENAMING
165
PRIMARY GLOBAL SERVER AND EACH SECONDARY GLOBAL SERVER UPDATE CURRENT VERSION OF SET OF FILES ON EACH CONTENT SERVER BY RENAMING
166
(OPTION) REENABLE ACCESS TO INDIVIDUAL LOCAL CONTENT SERVERS AFTER RENAMING IS COMPLETED
END

Fig. 3C

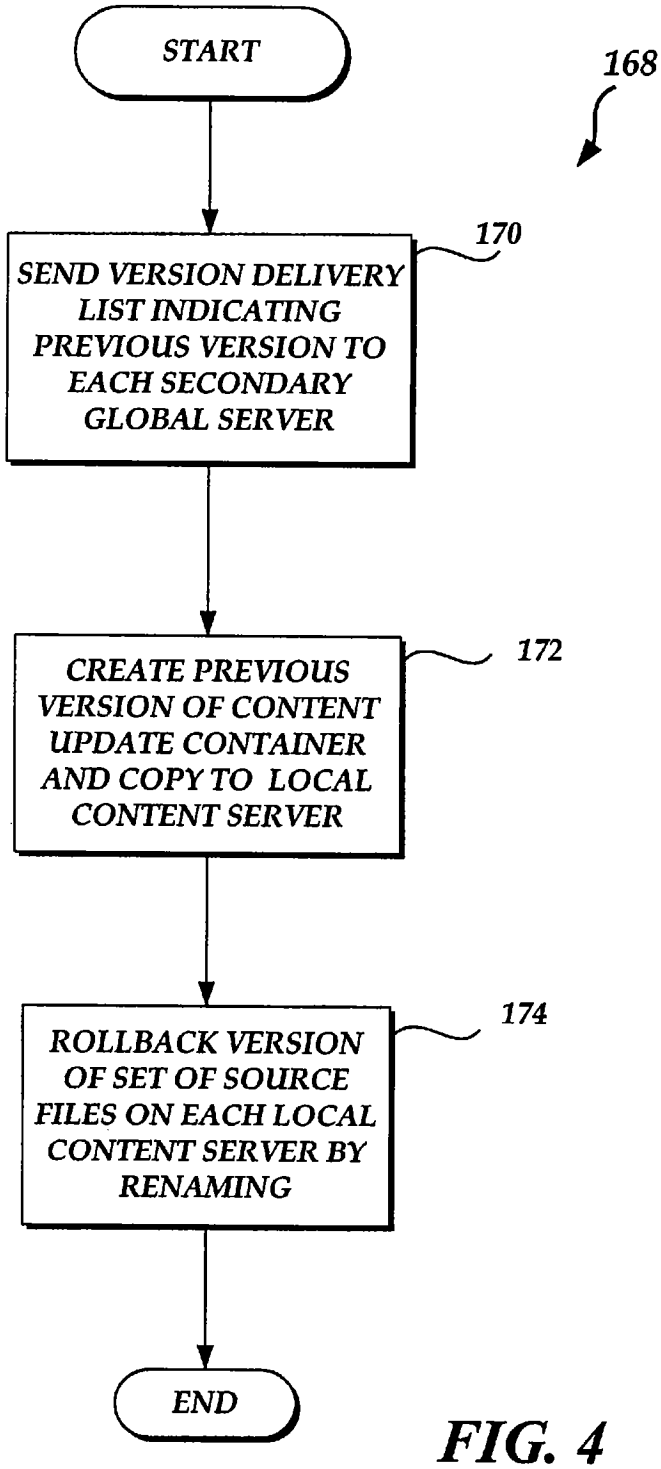

START
168
170
SEND VERSION DELIVERY LIST INDICATING PREVIOUS VERSION TO EACH SECONDARY GLOBAL SERVER
172
CREATE PREVIOUS VERSION OF CONTENT UPDATE CONTAINER AND COPY TO LOCAL CONTENT SERVER
174
ROLLBACK VERSION OF SET OF SOURCE FILES ON EACH LOCAL CONTENT SERVER BY RENAMING
END

FIG. 4

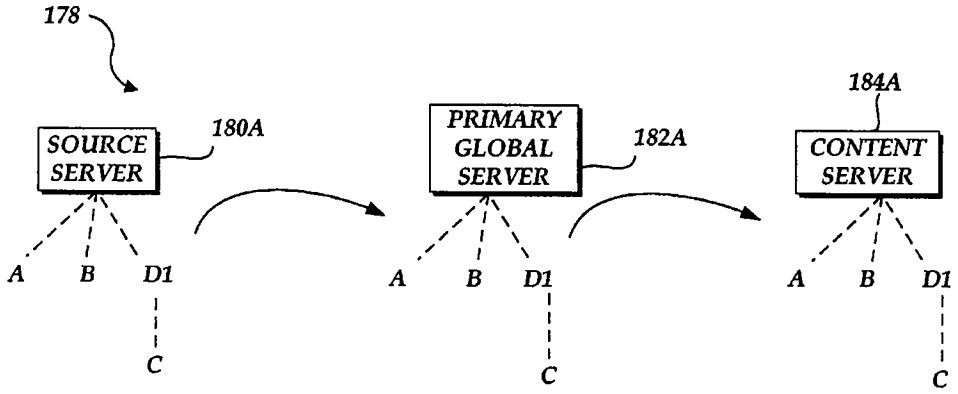

178
SOURCE SERVER
180A
A
B
D1
C
PRIMARY GLOBAL SERVER
182A
A
B
D1
C
184A
CONTENT SERVER
A
B
D1
C

FIG. 5A

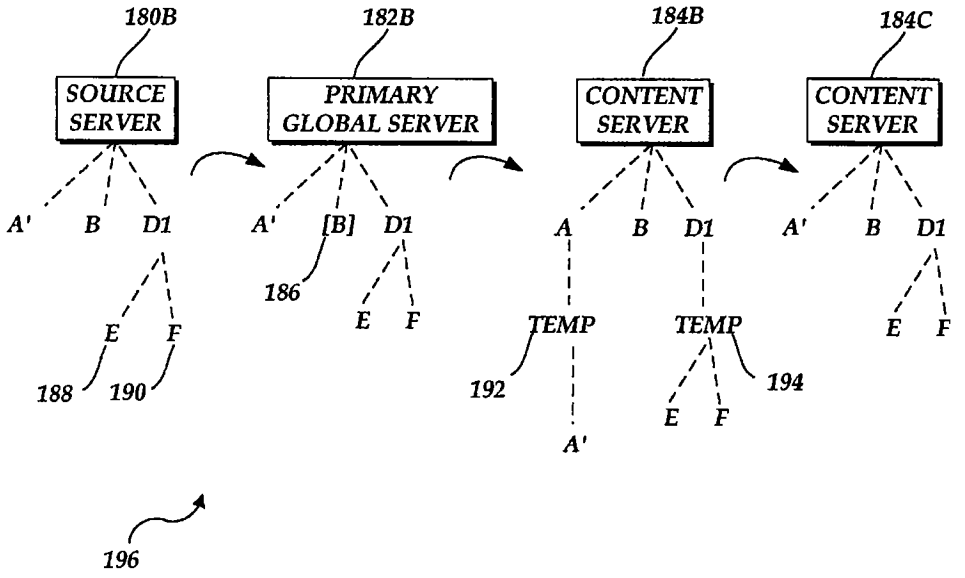

180B
SOURCE SERVER
A'
B
D1
E
F
188
190
182B
PRIMARY GLOBAL SERVER
A'
[B]
D1
186
E
F
184B
CONTENT SERVER
A
B
D1
TEMP
192
A'
TEMP
194
E
F
184C
CONTENT SERVER
A'
B
D1
E
F
196

FIG. 5B

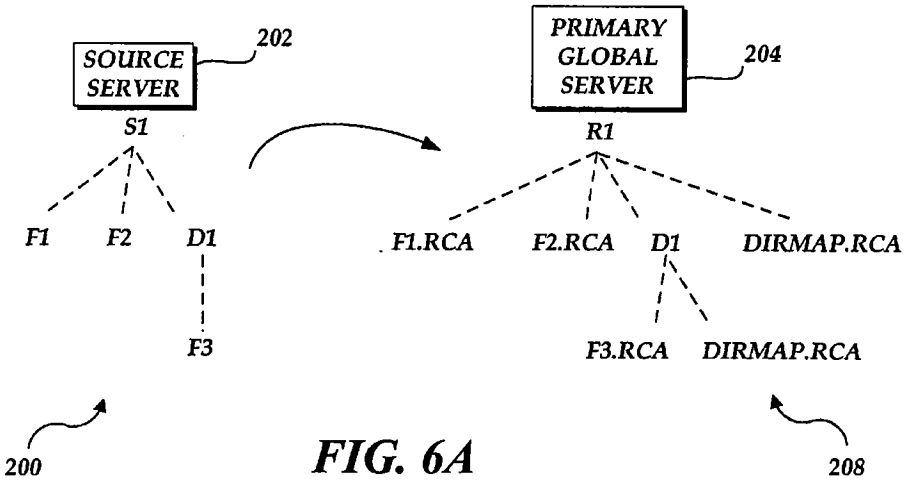

SOURCE SERVER
202
S1
F1
F2
D1
F3
PRIMARY GLOBAL SERVER
204
R1
F1.RCA
F2.RCA
D1
DIRMAP.RCA
F3.RCA
DIRMAP.RCA
200
208

FIG. 6A

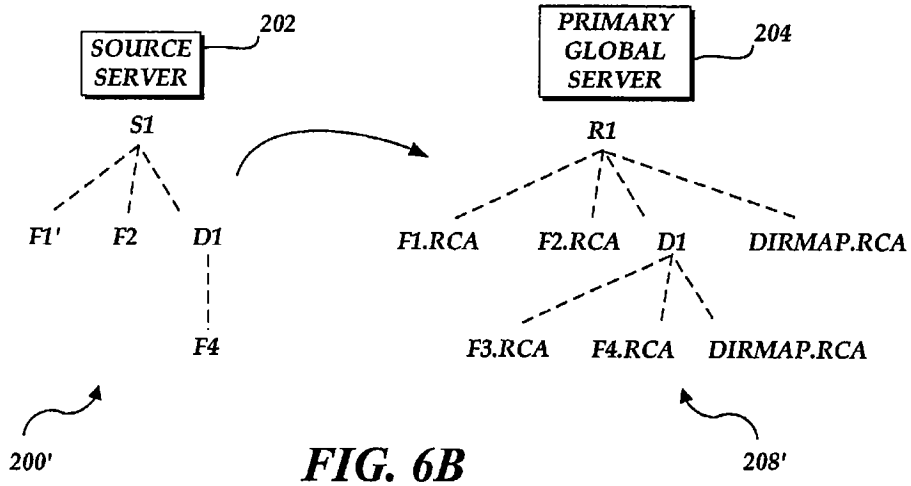

SOURCE SERVER
202
S1
F1'
F2
D1
F4
PRIMARY GLOBAL SERVER
204
R1
F1.RCA
F2.RCA
D1
DIRMAP.RCA
F3.RCA
F4.RCA
DIRMAP.RCA
200'
208'

FIG. 6B

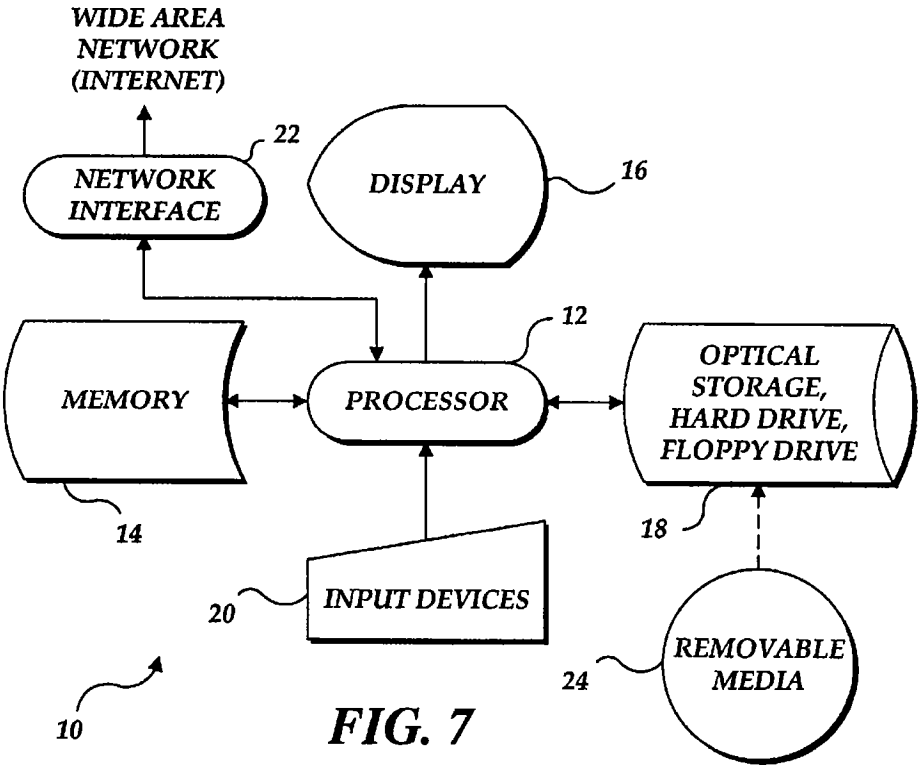

WIDE AREA NETWORK (INTERNET)
22
NETWORK INTERFACE
DISPLAY
16
12
MEMORY
PROCESSOR
OPTICAL STORAGE, HARD DRIVE, FLOPPY DRIVE
14
18
20
INPUT DEVICES
REMOVABLE MEDIA
24
10

FIG. 7

METHOD AND SYSTEM FOR AUTOMATICALLY UPDATING THE VERSION OF A SET OF FILES STORED ON CONTENT SERVERS

This application claims priority from Provisional application Ser. No. 60/140,213, filed Jun. 22, 1999.

FIELD OF THE INVENTION

This application relates generally to distributing updates to geographically distributed servers on a network, and, more specifically, to enabling the version of each source file stored on heterogeneous content servers to be automatically updated.

BACKGROUND OF THE INVENTION

Often, source files for web content servers are coded by multiple programmers on remotely located (stage) source servers. It is not unusual for one programmer(s) to code "HTML" files on one source server while another programmer(s) creates executable and/or image files on another source server. Once a programmer debugs a newly created/edited update file, it is eventually distributed to each content server and placed in a corresponding file directory. Historically, the distribution of the current version of a set of "updated" or new files from remotely located source servers through the Internet to content servers has proven to be a difficult task for several reasons. One reason is that the file directory structure and hardware configuration can vary between individual web content servers. In this case, the distribution of a set of files for each web content server must be separately organized according to each server's file directory structure and hardware capabilities. Another reason is that the actual size of the set of files may be so large that their distribution is relatively slow on a network with limited bandwidth capabilities.

Therefore, a need exists for a computer implementable method of distributing a set of the current version of source files to a plurality of content servers using a minimal amount of bandwidth. Preferably, the method will tailor the distribution of the set of source files according to the configuration, i.e., file structure and the hardware constraints, of each content server. Also, preferably the method would provide a facility for rolling back the current version of the set of source files to a previous version. The present invention is directed to providing such a computer implementable method.

SUMMARY OF THE INVENTION

In accordance with the present invention, a computer implementable method for updating a version of a set of source files stored on a content server over a network, comprising: (a) determining a configuration of each content server on the network, the configuration enabling a source file to be copied to a location on the content server; (b) identifying each source file on a source server that is different than any source file stored on a global server; (c) copying each identifiably different source file from the source server to the global server, each source file copied from the source server and a set of source files stored on the global server being employed to create a current version of the set of source files on the global server; and (d) employing the configuration of each content server to copy the current version of each source file that is included in the set of source files on the global server to a directory created on each content server, whereby the version of the set of source files stored on each content server is updated by renaming the current version of each source file copied to the directory on each content server.

In accordance with other aspects of the present invention, the method provides for renaming each current version of each source file that is copied to the directory created on each content server; and deleting the directory created on the content server and deleting another version of each source file that is updated by the renaming of the current version of each source file copied to the content server directory.

In accordance with yet other aspects of the present invention, the method provides for when the current version of each source file is copied to the directory created on each content server, disabling access to the set of source files on a particular content server until the renaming of the current version of each source file copied to the directory on the particular content is completed.

In accordance with still other aspects of the present invention, the method provides for when the current version of each source file is copied to the directory created on each content server, starting the renaming process with the current version of each copied source file that is furthest away from the root directory of each content server.

In accordance with other aspects of the present invention, the method provides for archiving each version of the set of source files in a repository on the global server, the archiving causing each source file to be individually compressed and stored as an archived object in the repository associated with the global server. The repository can be a versioned file tree repository for the set of source files.

In accordance with still further aspects of the present invention, the method provides for when a return to a previous version of the set of source files is requested, retrieving each archived object associated with the previous version of the set of source files from the repository associated with the global server. Each archived object associated with the previous version of the set of source files is unarchived to reconstitute each source file needed to upgrade the set of source files on the content server to the previous version. Each reconstituted source file is copied to a directory created on each content server, whereby the version of the set of source files on each content server is upgraded to the previous version by renaming the copied reconstituted source files.

In accordance with still other aspects of the present invention, the method provides for enabling a user to edit the configuration for each content server. Alternatively, the method may provide for automatically obtain the configuration for each content server.

In accordance with other aspects of the present invention, the method provides for employing a file access protocol to gain file level access to each source file, including FTP, NFS, CIFS and MFTP. The file access protocol may employ one port to send and receive data that includes a message and a source file. The type of source file includes image, hyper text mark-up language (HTML), script, sound, video, text, picture and application program code.

In accordance with yet other aspects of the present invention, the method provides for when a new content server is added to the network, employing the current version of the set of source files stored in a repository on the global server and a configuration of the new content server to replicate the current version of the set of source files in at least one directory created on the new content server.

In accordance with still further aspects of the present invention, the method provides for copying the differences

in the set of source files on the source server to a primary global server which generates a particular container that includes the differences in the set of source files stored on each remotely located secondary global server. The primary global server distributes the particular container from the primary global server to each associated secondary global server which employ the contents of the particular container to replicate the current version of the set of source files in a repository on the Secondary global server. The current version of each source file stored in the repository on the Secondary global server that is identified as necessary to replicate the current version of the set of source files on the content server is copied to another directory created on each content server that is local to the secondary global server. The set of source files may be stored in a versioned file tree repository on the primary global server and each secondary global server.

In accordance with yet other aspects of the present invention, the method provides for automatically distributing the container to the secondary global server. Alternatively, the distribution of the container to the secondary global server can be selectively enabled by an input. Also, the updating to the current version of the set of source files on the content server can be automatic or selectively enabled by an input. Additionally, each container can be distributed in a plurality of packets to the secondary global server and each packet may have a size that is less than a size of the container.

In accordance with other aspects of the present invention, the method provides for encrypting each message transmitted between the primary global server and each secondary global server.

In accordance with still other aspects of the present invention, the method provides for distributing a particular list to each secondary global server. The distributed list is employed by each secondary global server to identify the particular version for upgrading the set of sources files on each local content server.

In accordance with still other aspects of the present invention, the method provides for when another global server is added to the network, creating a copy of the versioned file tree repository for the set of source files. The versioned file tree repository for the set of source files is replicated on the other global server which employs the set of source files included in the versioned file tree repository to update the version of the set of source files stored on each content server that is local to the other global server. The type of the other global server may be primary or secondary.

In accordance with other additional aspects of the present invention, a system which implements substantially the same functionality in substantially the same manner as the methods described above is provided.

In accordance with yet other additional aspects of this invention, a computer-readable medium that includes computer-executable instructions that may be used to perform substantially the same methods as those described above is provided.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing aspects and many of the attendant advantages of this invention will become more readily appreciated as the same become better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings, wherein:

FIG. **1** illustrates an overview of the system architecture for implementing the present invention;

FIG. **2** is a flowchart showing an overview of the logic for updating files on remotely located content servers;

FIGS. **3A–3C** are flowcharts that illustrate in more detail the logic for updating files on remotely located content servers;

FIG. **4** is a flowchart showing the logic for rolling back a version of files on remotely located content servers;

FIG. **5A** is an overview of the file directory structure for an initial version of a set of source files that are created on a source server and copied to a Primary global server and a content server;

FIG. **5B** is an overview of the file directory structure for an updated version of the set of source files that are created on the source server and copied to the Primary global server and the content server;

FIG. **6A** is an overview of the initial versioning of a source tree that is created on the source server and copied to the Primary global server;

FIG. **6B** is an overview of the second versioning of a source tree that is modified on the source server and copied to the Primary global server; and

FIG. **7** illustrates an exemplary server computer system.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention provides for managing the distribution and synchronization of a set of updated content and application (source) files for remotely located heterogeneous content servers with reduced impact on a network's bandwidth. A particular set of source files for each content server is automatically updated according to the directory structure and hardware configuration of each content server. The present invention is typically employed with a Primary global server that is in communication with local source servers and local content servers. Also, the Primary global server may be used with at least one geographically separate Secondary global server that is in communication with other content servers that are local to the Secondary global server.

Generally, the new or changed (updated) set of source files for the content servers are created on the source servers. The Primary global server stores a representation of the source servers' current version of the set of updated source files either at determined intervals or at the direction of a user. The copied set of source files include the name, time stamp and size of each source file. The Primary global server also stores configuration files indicating the particular file directory structure and hardware constraints for each content server that is locally coupled to the Primary global server and each Secondary global server. The content server hardware constraints are usually entered manually into the Primary global server. Alternatively, the Primary and Secondary global servers may automatically determine access control and account information for each content server on the network.

The Primary global server generates a version delivery list for each Secondary global server that indicates a particular update version for each local content server. The version delivery list also includes the file directory structure and the hardware constraints of each local content server. The Primary global server also generates a version change container for each Secondary global server based on its path and the difference between the updated version of the set of source files stored in a versioned file tree repository on the Primary global server and the current version of the set of source files stored in a another versioned file tree repository on each

Secondary global server. At a determined interval, the version delivery lists and the version change containers are distributed from the Primary global server to each Secondary global server. Also, each Secondary global server employs the set of source files included with the version change container to update the version of the set of source files stored in the Secondary global server's versioned file tree repository.

The Primary global server and each Secondary global server generate a current version of the set of new and/or changed source files and files to be removed for each local content server based on the update version identified in the version delivery list. The Primary and Secondary global server generate a content change container that includes the current version of the set of new and/or changed source files and indicates which source files are to be removed on each local content server. Employing the contents of the content change container, each Secondary global server will make the update version changes as indicated in the version delivery list by copying the update version of the set of new and/or changed source files to temporary sub-directories on each of their associated local content servers. Similarly, the Primary global server will copy the update version of the set of new and/or changed source files to temporary sub-directories on each of its associated local content servers. The Primary and Secondary global servers change the version of the set of source files on the local content servers by renaming the update version of the set of source files copied to the temporary sub-directories.

Additionally, when a user indicates that a current version of the set of source files on the local content servers should be rolled back to a previous version, the Primary global server creates a "rollback" version delivery list that is provided to each Secondary global server. A Secondary global server employs the previous version indicated in the rollback version delivery list to generate the previous version of the set of source files necessary to restore the previous version of the set of source files. These source files are copied to a temporary sub-directory on each local content server. Similarly, the Primary global server generates the previous version of the set of source files and copies these files to a temporary sub-directory on each local content server.

The Primary and Secondary global servers rollback to the previous version by renaming the set of source files copied to the temporary sub-directories on the local content servers. Additionally, when the set of source files are copied to the sub-directories on the local content servers, the Primary and Secondary global servers will delete source files that did not exist in the previous version and they may temporarily disable user access to the local content servers until the previous version of the set of source files are renamed.

Encryption may be provided for all communication between the Primary Global server, Secondary global server(s) and the local content servers. Also, file compression may be provided for the distribution of version change containers between the Primary global server and the Secondary global server(s). The present invention may employ any file access method to gain file level access to a source file on a server including a file transport protocol (FTP), network file system (NFS), computer interconnect file system (CIFS) and multi-cast file transfer protocol (MFTP).

System Overview

FIG. **1** illustrates an overview **100** of the present invention employed in a network environment that includes a wide area network such as the Internet **101**, FIG. **1** includes a data center **103** coupled to the Internet **101**. The data center **102** includes source servers **112A**, **112B** and **112C** for creating file based content and applications, e.g., HTML pages, graphic image format (GIF) images and executables and a test server **114** for testing new and changed source files. The data center **103** also includes a Primary global server **102** in communication with an optional firewall server **106A**, local content servers **104A**, a server array controller **105A** and a manual entry device **110**. The Primary global server **102** is connected to the Internet **101** (optionally through firewall server **106A**) and is in communication with the source servers **112A–C** and the test server **114**.

The manual entry device **110** enables a user to provide information for the Primary global server **102** including server configuration, file distribution profiles, hardware constraints and set up rules. The Primary global server **102** provides the current version of a set of source files to the local content servers **104A**. The server array controller **105A** manages access to the information, e.g., content and applications, on the content servers **104A**. Typically, a server array controller manages a pool of redundant content (node) servers to provide access to requested resources such as a BIG/ip™ server array controller available from F5 Networks, Inc., Seattle, Wash.

The Primary global server **102** distributes containers to Secondary global servers **108A** and **108B** across the Internet **101**. The Secondary global servers **108A** and **108B** form part of geographically separate data centers **109** and **111**. The Secondary global servers **108A** and **108B** are shown coupled through optional firewall servers **106B** and **106C**, respectively, to the Internet **101**. Each of the Secondary global servers **108A** and **108B** are in communication with one of more local content servers **104B** and **104C**, respectively. As a result, the Secondary global servers **108A** and **108B** can provide a current version of a set of source files to their associated local content servers **104B** and **104C**. Each geographically separate data center **109** and **111** also includes a server array controller **105B** and **105C** to manage access to the content and applications on the local content servers **104B** and **104C**. FIG. **1** should be considered exemplary, not limiting. If desired, one or more than two, geographically separated data centers may be included in a network employing the present invention.

In another embodiment, the present invention can be employed to provide updates to a content server that is not managed by a server array controller. Additionally, the Primary global server can implement the present invention without the use of Secondary global servers at geographically separate data centers, such an embodiment of the invention would be employed when all of the content servers are local to the data center that includes the Primary global server.

Flowcharts

FIG. **2** is a flow chart illustrating an overview **116** of the main logic for providing a current version of a set of source files from at least one source server to a plurality of content servers. Moving from a start block, the logic steps to a block **118** where configuration information from each content server is determined e.g., paths, file directory structure and hardware constraints. The determination can be made by recording entered configuration information (hardware and software). Alternatively, the Primary and Secondary global servers may automatically read the configuration information of each local content server. The configuration information may be provided out of band to each Primary and Secondary global server when new configuration information becomes available and/or at determined intervals.

The logic flows to a block **120** where the set of source files created on the source servers are identified according to

name, size and date of creation/modification. A Primary global server copies only those source files from the source servers that are determined to be different than the set of source files stored in the versioned file tree repository on the Primary global server. As a result, the present invention employs differences to identify the source files that are to be copied from the source servers to the Primary global server.

Advancing to a block **122**, the Primary global server creates a particular version delivery list for each Secondary global server. The version delivery list indicates the version upgrade for each set of source files on each local content server.

Also, the Primary global server creates a version change container based on the difference between the current version of the set of source files stored in the versioned file tree repository on the Primary global server and the version of the set of source files stored in another versioned file tree repository on each Secondary global server. The version change container references the names of all of the source files that are included in or deleted from the current version of the set of source files. The version change container also includes the actual file data for each new source file and a portion of the file data for each existing source file that was modified in the current version of the set of source files.

After the creation of the version delivery lists and the version change container, the Primary global server provides copies of the version change container and the particular version delivery list on each Secondary global server. It is understood that the Primary global server stores a copy of each version of a source file from a source server that is determined to be different than the version of the source file on the Primary global server. Alternatively, each Secondary global server stores a copy of each version of a source file that is provided in a version change container from the Primary global server in the Secondary global server's versioned file tree repository.

The logic steps to a block **124** where the Primary global server and each Secondary global server create a content change container for each local content server and copy new and/or changed source files to at least one sub-directory on the corresponding local content server. For each Primary and Secondary global server, the copied source files are based on previously determined configuration information for a particular local content server and the version of the set of source files identified in the version delivery list. Since the present invention "assumes" that a previously copied source file on a content server is persistent, another copy of a previously copied and unchanged version of a source file is not included in the set of the current version of source files that are copied to a sub-directory on the local content server, i.e., the Primary and Secondary global servers copy the actual file data for the current version of new and modified source files to sub-directories on local content servers.

The logic flows to a block **126** where the Primary and Secondary global servers update the version of the set of source files on each local content server by renaming the source files copied to a sub-directory on each local content server. Also, any previously copied source files that were removed from the current version of the set of source files are deleted on each content server. When the renaming and/or deleting is completed, the logic will move to an end block and terminate.

FIGS. **3A–3C** form a flowchart **130** that shows in greater detail the logic of the present invention. Starting with FIG. **3A**, the logic moves from a start block and steps to a block **132** where the Primary global server queries a database that stores information about content servers coupled to the network and uses the results of the query to build a list of content servers and their hardware/software configuration. The logic flows to a block **134** where the Primary global server uses the results of another query of the database to build a list of the available source servers and their respective paths.

The logic advances to the block **136** where the Primary global server gains file level access to each source server with an FTP connection. The logic steps to a block **138** where the Primary global server examines the source (content and application) files on each source server and identifies each new and/or modified source file by comparing the name, time stamp and size of each source file on each source server to the current version of each source file stored on the Primary global server in a versioned file tree.

When a source file with the same name exists on both a source server and the Primary global server, the present invention identifies the most current version by comparing their sizes and time stamps. If the sizes of the source files with the same name are different or the time stamp of the source file on the source server is different than the time stamp of the Primary global server's source file, the source server's source file is identified as the most current version. Further, when another source file with the same name is not on the Primary global server, the source file on the source server is identified as the current version. Also, when a named source file only exists on a Primary global server, this source file is not identified as a member of the current version of the set of source files.

The logic flows to a block **140** where the Primary global server gains file level access (FTP connection) to each source server that includes a source file that is identified as different than the current version of that particular file in the versioned file tree on the Primary global server. Each identified source file is copied to a new version in the versioned file tree repository on the Primary global server.

The Primary global server calls a library, e.g., the Revision Control Engine (RCE), to store file level differences between the current and previous versions of each source files. A discussion of FIGS. **6A** and **6B** below presents the functionality of the versioned file tree repository in greater detail.

In another embodiment, another file access protocol may be employed to transfer information, e.g., files, messages and data, between the Primary, Secondary, source and content servers. This other protocol could use a single port to enable all of the functions of the present invention, such as enabling the Primary global server to control the operation of the Secondary global server.

The logic moves to a block **142** where the Primary global server generates version delivery lists and a list of Secondary global servers and their respective paths. Also, the Primary global server generates a version change container for each Secondary global server that may include a reference value associated with the current version of the set of source files.

Turning to FIG. **3B** from FIG. **3A**, the logic steps to a block **146** where the Primary global server archives (compresses) each version change container. A third party facility may be used to implement a tape archive (TAR) command to compress each version change container. The logic moves to a block **148** where a copy of the archived version change container is encrypted and transmitted to each Secondary global server. To reduce any adverse impact on the bandwidth capacity of the network, each version change container may be broken down into relatively small units that are individually encrypted and transmitted to a Secondary global server.

The logic moves to a block **150** where the Primary global server sends an encrypted message to each Secondary global server to unarchive the version change container. The logic steps to a block **152** where each Secondary global server unarchives the relatively small transmitted units and copies each unarchived source file to a new version in the versioned file tree repository on each Secondary global server.

The logic flows to a block **154** where the Primary global server sends a version delivery list to each Secondary global server. In this case, the version delivery list indicates the current version, however, it should be appreciated that this list could indicate a previous version of the set of source files.

The logic flows to a block **156** where the Primary global server and the Secondary global server build a content update container for each local content server that includes the actual file data (new source files and modified portions of previously existing source files) and indicate each source file to be deleted from the content server. The content update container is based on the two versions identified in the version delivery list. The logic advances to a block **158** where the Primary global server sends an encrypted message to each Secondary global server to copy the new and/or modified source files in the content update container to at least one sub-directory on each local content server.

Moving from FIG. **3**B to FIG. **3**C, the logic steps to a block **160** where the Primary global server and each Secondary global server gain file level access to the file directory on each local content server and copy the new and/or source files to a sub-directory on each local content server.

Optionally, the logic may move to a block **162** where the Primary and each Secondary global server will disable access to a local content server until the renaming of the current version of the set of source files is completed. In another embodiment, the present invention may start renaming source files from the "bottom" up of a local content server's file directory and may not disable access to the local content server during the copying/renaming process. It is envisioned that the Primary global server may provide a separate encrypted message to each Secondary global server to disable access to the local content servers during the renaming process.

The logic advances to a block **164** where the Primary global server sends an encrypted message to each Secondary global server to update the version of the set of source files stored on each local content server by renaming the actual source file data copied to a sub-directory on each local content server.

At block **165**, the Primary and Secondary global servers update the version of the set of source files on each local content server by renaming. A previous version of an individual source file and a deleted source file are removed when the current version of the set of source files are renamed.

Optionally, the logic steps to a block **166** where each Secondary global server will re-enable access to each local content server disabled for the renaming. Also, it is envisioned that the Primary global server may provide a separate encrypted message to each Secondary global server for enabling access to the local content servers after the renaming process is completed. Next, the logic flows to an end block and terminates.

In FIG. **4**, a flow chart is shown illustrating an overview **168** of the logic for "rolling back" the current version of the set of source files stored on local content servers to a previous version. Advancing from a start block, the logic moves to a block **170** where the Primary global server sends a version delivery list to each Secondary global server indicating a previous version of the set of source files stored in a versioned file tree repository on the Secondary global server.

The logic steps to a block **172** where the Primary and Secondary global servers generate a content update container that includes a previous version of the set of source files for each local content server. The Primary and Secondary global servers copy the previous version of modified source files and restore removed source files from the previous version to at least one sub-directory on the local content servers. The logic flows to a block **174** where the Primary and Secondary global servers cause the version of the set of source files on each local content server to roll back by renaming the previous version of the set of source files included in the content update container copied to the a sub-directory on each local content server. Also, any version of the source files that are newer than the previous version are deleted at this time. Next, the logic advances to an end block and terminates.

Although not shown, the present invention may be employed to rollback or increase more than one version of the set of source files at a time. For example, when one content server has a first version of the set of source files and other content servers have the second version of these source files, the present invention will separately update the first version to the second version before updating every content server to the third version of the set of source files.

The present invention is relatively fault tolerant because each (Primary and Secondary) global server can store redundant copies of all of the information stored in the repositories of every other server, e.g., several previous versions of the set of source files. If the Primary global server or any one of the Secondary global servers should fail, the related information can be provided to a replacement (Primary or Secondary) global server from the information stored in a versioned file tree repository on any one of the other operational global servers.

Data Structures

FIG. **5**A illustrates an overview **178** of the file directory structure for a first version of the set of source files that is distributed from a source server **180**A to a Primary global server **182**A and a content server **184**A. For all three of these servers, files "A" and "B" are shown one level below the root directory and file "C" is shown below the "D1" sub-directory root.

FIG. **5**B shows an overview **196** of the file directory structure at each server when a second version of the set of source files is copied from a source server **180**B to a Primary global server **182**B and then to a content server **184**B. At the source server **180**B, the file directory structure of the second version of the set of source files is substantially similar to the first version shown in FIG. **5**A except that the "C" file is deleted and new source files "D" and "E" are disposed below the "D1" sub-directory root. Also, the second version of the set of source files includes a modified source file "A'."

At the Primary global server **182**B, the file directory structure of the second version of the set of source files is substantially similar to the second version of the set of source files stored at the source server **180**B. However, since source file "B" did not change between the first and second versions of the set of source files, the second version includes a reference value **186** indicating that source file "B" in the first version is to be reused in the second version of the set of source files. As a result, the actual size of subsequent versions of the set of source files may be reduced by referencing unchanged source files that were previously stored on the Primary global server **182**B.

Additionally, prior to the renaming method discussed in greater detail above, the file directory structure of the second version of the set of source files on content server **184**B is substantially similar to the second version stored at the source server **180**B. Except that under the root directory a temporary sub-directory **192** was created for the changed source file "A'." Also, a temporary sub-directory **194** was created under sub-directory root "D1" for the new files "E" and "F."

Content server **184**C shows the second version of the file directory structure for the set of source files after renaming has occurred. The temp directory **192** is deleted and source file "A" has replaced the previous version source file "A." Also, the temp directory **194** is deleted and the new source files "E" and "F" are under the "D1" sub-directory.

In FIG. 6A, a file tree **200** representing a set of source files on a source server **202** is shown. Directly below an "S1" root directory, two source files "F1" and "F2" are positioned along with a "D1" sub-directory which is a root for a source file "F3." Further, each source file in the file tree **200** is represented in a versioned file tree repository **208** of RCE archived source files with an RCA file extension. However, it is understood that other types of libraries may be employed with the present invention to archive a source file and produce an archived source file with another file extension.

In the versioned file tree repository **208**, two RCE archived source files "F1.RCA" and "F2.RCA," a sub-directory "D1" and a directory map "DIRMAP.RCA" are located below an "R1" root directory, i.e., R1/F1 RCA, R1/F2.RCA, R1/D1 and R1/DIRMAP.RCA. Also, an archived source file "F3.RCA" and a directory map "DIRMAP.RCA" are disposed below the "D1" sub-directory level, i.e., R1/D1/F3.RCA and R1/D1/DIRMAP.RCA.

Each level of the versioned file tree repository **208** includes an RCE archived directory map file named DIRMAP.RCA. For each version of the set of source files copied from the source servers and archived on the Primary global server, the directory map file includes the version, size and time stamp for each RCE archived source file and sub-directory at the same directory level in the versioned file tree repository **208** as the particular directory map file. Also, for the top level directory map file, the present invention generates an alias name that maps a particular version of the set of RCE archived source files to the actual version of the set of source files that are provided to the local content servers.

For example, when the initial version of the actual set of source files is provided to the local content servers, the top level directory map (R1/DIRMAP.RCA) will include a versioned list that maps the initial version value ("1.1") to a set of RCE archived source files and an alias name. In this case, the list for R1/DIRMAP.RCA would include <F1, 1.1>, <F2, 1.1>, <D1, 1.1.>and <V1, 1.1>. Similarly, the list for the "D1" sub-directory map file (R1/D1/DIRMAP.RCA) would include <F3, 1.1>. It is to be appreciated that only the top level directory map file contains an alias name ("V1") to map the actual version of the set source files provided to the content servers to the version of the RCE archived set of source files on the Primary global server.

FIG. 6B shows a modified file tree **200'** for a second version of the set of source files created on the source server **202**. Directly below the "S1" root directory is disposed a modified source file "F1'," the previously existing and unchanged source file "F2" and the "D1" sub-directory for a new source file "F4." A modified versioned file tree repository **208'** for the set of RCE archived source files is located on the Primary global server **204** below the "R1" root directory which includes the modified RCE archived source file "F1.RCA," a directory map "DIRMAP.RCA," the unchanged RCE archive source file "F2.RCA" and a sub-directory "D1." Also, below the "D1" sub-directory level is disposed the previously existing RCE archive source file "F3.RCA" that is deleted from the second version of the set of source files on the source server **202**, a new RCE archived source file "F4.RCA" and another directory map "DIRMAP.RCA."

The RCE library provides for automatically incrementing the version of new and changed archived source files. In this case, the second version value ("1.2") is automatically associated with the changed RCE archived source file "F1.RCA and the new RCE archived source file "F4.RCA." Also, the alias name of "V2" is mapped to the RCE archived source files associated with the second version value ("1.2"). In this exemplary embodiment, the top level directory map file (R1/IDIRMAP.RCA) contains a list that associates first and second version values with RCE archived source files, sub-directories and alias names, e.g., the R1/DIRMAP.RCA list contains <F1, 1.2>, <F2, 1.1>, <D1, 1.2.>, <V1, 1.1>and <V2, 1.2>. It is further envisioned that each modified RCE archived source file will contain every previous version of the file, e.g., "F1.RCA" would include the 1.1 and the 1.2 versions of the RCE archived source file. Similarly, the sub-directory directory map file (R1/D1/DIRMAP.RCA) would contain a list that includes <F3, 1.1> and <F4, 1.2>.

It is important to note that the alias names ("V1" and "V2") in the top level directory map file are used to reference all of the new or changed files for each version in the versioned file repository of the set of RCE archived source files. In this way, the present invention can employ the alias names to support duplicate versioned file tree repositories when the version sequence for updating a set of source files is not identical for every content server. Also, the use of an alias name enables the present invention to only touch/access the new/changed RCE archived source files and directories when updating the version of a set of source files on a local content server.

In the example discussed above, the first and second versions ("1.1" and "1.2") of the actual set of source files provided to the local content servers were associated with the alias names "V1" and "V2," respectively. However, it is envisioned that a subsequent version upgrade to the actual set of source files provided to a local content server might skip a version that is RCE archived on the Primary global server. For example, a fourth version of the set of RCE archived source files could be employed to provide the third version upgrade to the set of source files on the local content servers. In this case, an alias name of "V4" would be mapped to the third version upgrade ("1.3") of the set of RCE archived source files stored in a versioned file tree repository on the Secondary global server.

System Configuration

FIG. **7** is a pictorial diagram of a Primary global server **10** suitable for executing an application program embodying the present invention. FIG. **7** shows a processor **12** coupled bi-directionally to a memory **14** that encompasses read only memory (ROM) and random access memory (RAM). ROM is typically used for storing processor specific machine code necessary to bootup the computer comprising the Primary global server **10**, to enable input and output functions, and to carry out other basic aspects of its operation. Prior to running any application program, the machine language code comprising the program is loaded into RAM within memory **14** and then executed by processor **12**. Processor **12**

is coupled to a display **16** on which the visualization of an HTML response discussed above is presented to a user. Often, programs and data are retained in a nonvolatile memory media that may be accessed by a compact disk-read only memory (CD-ROM) drive, compact disk-read/write memory (CD-R/W) drive, optical drive, digital versatile disc (DVD) drive, hard drive, tape drive and floppy disk drive, all generally indicated by reference numeral **18** in FIG. **7**. A network interface **22** couples the processor **12** to a wide area network such as the Internet.

As noted above, embodiments of the present invention can be distributed for use on the computer system for the Primary global server **10** as machine instructions stored on a memory media such as a floppy disk **24** that is read by the floppy disk drive. The program would then typically be stored on the hard drive so that when the user elects to execute the application program to carry out the present invention, the machine instructions can readily be loaded into memory **14**. Control of the computer and selection of options and input of data are implemented using input devices **20**, which typically comprise a keyboard and a pointing device such as a mouse (neither separately shown). Further details of the system for the Primary global server **10** and of the computer comprising it are not illustrated, since they are generally well known to those of ordinary skill in the art. Additionally, computer systems for a Secondary global server and the content server could be configured in substantially the same way as the computer system for the Primary global server **10** illustrated here, albeit different in other ways.

It is to be understood that embodiments of the present invention can be created to support all file based content and applications including GIF, TIFF, AVI, JPEG, MPEG, HTML pages, JAVA scripts, Active Server pages, postscript document format (PDF), ActiveX, JAVA, and application programs. It is envisioned that embodiments of the present invention provides security mechanisms for protecting the delivery of content and application files to content servers. These security mechanisms enable remote administration of the present invention through a secure shell command line (SSH) and a secure socket layer (SSL) for browser based administration.

It is envisioned that embodiments of the present invention will enable a new content server to be deployed with minimal effort. A Primary or Secondary global server can employ the contents of the most current update file tree object to automatically generate a current version of the set of source files for a new local content server. Additionally, an important aspect of the present invention is that proprietary software does not have to be installed on the source servers or content servers to receive the benefits of the present invention.

While the preferred embodiment of the invention has been illustrated and described, it will be appreciated that various changes can be made therein without departing from the spirit and scope of the invention.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. Method for updating a version of a set of source files stored on a content server over a network, comprising:

(a) determining a configuration of each content server on the network, the configuration enabling a source file to be copied to a location on the content server;

(b) identifying each source file on a source server that is different than any source file stored on a global server;

(c) copying each identifiably different source file from the source server to the global server, each source file copied from the source server and a set of source files stored on the global server being employed to create a current version of the set of source files on the global server; and

(d) employing the configuration of each content server to copy the current version of each source file that is included in the set of source files on the global server to a directory created on each content server, whereby the version of the set of source files stored on each content server is updated by renaming the current version of each source file copied to the directory on each content server.

2. The method of claim **1**, further comprising:

(a) renaming each current version of each source file that is copied to the directory created on each content server; and

(b) deleting the directory created on the content server and deleting another version of each source file that is updated by the renaming of the current version of each source file copied to the directory.

3. The method of claim **2**, wherein when the current version of each source file is copied to the directory created on each content server, further comprising disabling access to the set of source files on a particular content server until the renaming of the current version of each source file copied to the directory on the particular content is completed.

4. The method of claim **2**, wherein when the current version of each source file is copied to the directory created on each content server, further comprising starting the renaming process with the current version of each copied source file that is furthest away from the root directory of each content server.

5. The method of claim **1**, further comprising archiving each version of the set of source files in a repository on the global server, the archiving causing each source file to be individually compressed and stored as an archived object in the repository associated with the global server.

6. The method of claim **5**, further comprising:

(a) when a return to a previous version of the set of source files is requested, retrieving each archived object associated with the previous version of the set of source files from the repository associated with the global server;

(b) unarchiving each archived object associated with the previous version of the set of source files to reconstitute each source file needed to upgrade the set of source files on the content server to the previous version; and

(c) copying each reconstituted source file to a directory created on each content server, whereby the version of the set of source files on each content server is upgraded to the previous version by renaming the copied reconstituted source files.

7. The method of claim **5**, wherein the repository is a versioned file tree repository for the set of source files.

8. The method of claim **1**, further comprising enabling a user to edit the configuration for each content server.

9. The method of claim **1**, further comprising automatically obtaining the configuration for each content server.

10. The method of claim **1**, further comprising employing a file access protocol to gain file level access to each source file, including FTP, NFS, CIFS and MFTP.

11. The method of claim **10**, wherein employing the file access protocol to gain file level access to each source file further comprises employing one port to send and receive data that includes a message and a source file.

12. The method of claim 1, wherein a type of the source file includes image, hyper text mark-up language (HTML), script, sound, video, text, picture and application program code.

13. The method of claim 1, further comprising when a new content server is added to the network, employing the current version of the set of source files stored in a repository on the global server and a configuration of the new content server to replicate the current version of the set of source files in at least one directory created on the new content server.

14. The method of claim 1, further comprising:

(a) copying each identifiably different source file from the source server to a primary global server, the primary global server generating a separate container for each secondary global server, each container including the differences between the current version of the set of source files stored on the primary global server and a set of source files stored on each secondary global server associated with the container;

(b) distributing each container from the primary global server to each associated secondary global server, each secondary global server employing the contents of the container to replicate the current version of the set of source files in a repository on the secondary global server; and

(c) copying the current version of each source file stored in the repository on the secondary global server that is identified as necessary to replicate the current version of the set of source files on the content server to another directory created on each content server that is local to the secondary global server.

15. The method of claim 14, further comprising encrypting each message transmitted between the primary global server and each secondary global server.

16. The method of claim 14, further comprising storing the set of source files in a versioned file tree repository on the primary global server and each secondary global server.

17. The method of claim 16, further comprising:

(a) when another global server is added to the network, creating a copy of the versioned file tree repository for the set of source files; and

(b) replicating the versioned file tree repository for the set of source files on the other global server, the other global server employing the set of source files included in the versioned file tree repository to update the version of the set of source files stored on each content server that is local to the other global server.

18. The method of claim 17, wherein a type of the other global server includes primary and secondary.

19. The method of claim 14, wherein the distribution of the container to the secondary global server is automatic.

20. The method of claim 14, wherein the distribution of the container to the secondary global server is selectively enabled by an input.

21. The method of claim 14, wherein the updating to the current version of the set of source files on the content server is automatic.

22. The method of claim 14, wherein the updating to the current version of the set of source files on the content server is selectively enabled by an input.

23. The method of claim 14, further comprising distributing each container in a plurality of packets to the secondary global server, each packet having a size that is less than a size of the container.

24. The method of claim 14, further comprising distributing a particular list to each secondary global server, the list being employed by each secondary global server to identify the particular version for upgrading the set of sources files on each local content server.

25. A system for updating a set of source files on a remotely located content server over a network, comprising:

(a) a global server, comprising:

(i) a memory for storing logical instructions;

(ii) a network interface for communicating over the network; and

(ii) a processor for executing the logical instructions stored in the memory, the execution of the logical instructions causing functions to be performed, including:

(A) determining a configuration of each content server on the network, the configuration enabling a source file to be copied to a location on the content server;

(B) identifying each source file on a source server that is different than any source file stored on a global server;

(C) copying each identifiably different source file from the source server to the global server, each source file copied from the source server and a set of source files stored on the global server being employed to create a current version of the set of source files on the global server; and

(D) employing the configuration of each content server to copy the current version of each source file that is included in the set of source files on the global server to a directory created on each content server, whereby the version of the set of source files stored on each content server is updated by renaming the current version of each source file copied to the directory on each content server.

26. A computer-readable medium having computer-executable instructions for performing logical instructions stored in the medium, the execution of the logical instructions functions to be performed, comprising:

(a) determining a configuration of each content server on the network, the configuration enabling a source file to be copied to a location on the content server;

(b) identifying each source file on a source server that is different than any source file stored on a global server;

(c) copying each identifiably different source file from the source server to the global server, each source file copied from the source server and a set of source files stored on the global server being employed to create a current version of the set of source files on the global server; and

(d) employing the configuration of each content server to copy the current version of each source file that is included in the set of source files on the global server to a directory created on each content server, whereby the version of the set of source files stored on each content server is updated by renaming the current version of each source file copied to the directory on each content server.

27. The method of claim 1, further comprising obtaining the configuration for each content server when the configuration is changed.

28. The method of claim 1, wherein obtaining the configuration of each content server occurs at a determined interval.

29. The method of claim 1, wherein the copying of the current source files on the global server to the directory created on each content server occurs at a determined interval.

30. The method of claim 14, further comprising distributing each container from the primary global server to each associated secondary global server at a determined interval.

31. The system of claim 25, the execution of the logical instructions causing function to be performed, further including obtaining the configuration of each content server on the network when the configuration is changed.

32. The system of claim 25, wherein obtaining the configuration of each content server occurs at a determined interval.

33. The system of claim 25, wherein the copying of the current source files on the global server to the directory created on each content server occurs at a determined interval.

34. The computer-readable medium of claim 26, the execution of the logical instructions functions to be performed, further comprising obtaining the configuration for each content server when the configuration is changed.

35. The computer-readable medium of claim 26, wherein obtaining the configuration of each content server occurs at a determined interval.

36. The computer-readable medium of claim 26, wherein the copying of the current source files on the global server to the directory created on each content server occurs at a determined interval.

37. A method for updating a version of a set of source files stored on a content server over a network, comprising:

(a) determining a configuration of each content server on the network out of band when the configuration is changed, the configuration enabling a source file to be copied to a location on the content server;

(b) identifying each source file on a source server that is different than any source file stored on a global server;

(c) copying each identifiably different source file from the source server to the global server, each source file copied from the source server and a set of source files stored on the global server being employed to create a current version of the set of source files on the global server; and

(d) employing the configuration of each content server to copy the current version of each source file that is included in the set of source files on the global server to a directory created on each content server at a determined interval, whereby the version of the set of source files stored on each content server is updated by renaming the current version of each source file copied to the directory on each content server.

38. The method of claim 37, further comprising:

(a) copying each identifiably different source file from the source server to a primary global server, the primary global server generating a separate container for each secondary global server, each container including the differences between the current version of the set of source files stored on the primary global server and a set of source files stored on each secondary global server associated with the container;

(b) distributing each container from the primary global server to each associated secondary global server at the determined interval, each secondary global server employing the contents of the container to replicate the current version of the set of source files in a repository on the secondary global server; and

(c) copying the current version of each source file stored in the repository on the secondary global server that is identified to replicate the current version of the set of source files on the content server to another directory created on each content server that is local to the secondary global server.

39. The method of claim 37, wherein the determining of the configuration occurs at a predetermined interval.

40. A modulated data signal having computer-executable instructions, the execution of the computer-executable instructions causing actions, comprising:

(a) determining a configuration of each content server on the network when the configuration is changed, the configuration enabling a source file to be copied to a location on the content server;

(b) identifying each source file on a source server that is different than any source file stored on a global server;

(c) copying each identifiably different source file from the source server to the global server, each source file copied from the source server and a set of source files stored on the global server being employed to create a current version of the set of source files on the global server; and

(d) employing the configuration of each content server to copy the current version of each source file that is included in the set of source files on the global server to a directory created on each content server at a determined interval, whereby the version of the set of source files stored on each content server is updated by renaming the current version of each source file copied to the directory on each content server.

41. The modulated data signal of claim 40, the execution of the computer-executable instructions causing actions, further comprising:

(a) copying each identifiably different source file from the source server to a primary global server, the primary global server generating a separate container for each secondary global server, each container including the differences between the current version of the set of source files stored on the primary global server and a set of source files stored on each secondary global server associated with the container;

(b) distributing each container from the primary global server to each associated secondary global server at the determined interval, each secondary global server employing the contents of the container to replicate the current version of the set of source files in a repository on the secondary global server; and

(c) copying the current version of each source file stored in the repository on the secondary global server that is identified to replicate the current version of the set of source files on the content server to another directory created on each content server that is local to the secondary global server.

42. The modulated data signal of claim 40, wherein determining the configuration of each content server occurs out of band at a determined interval.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

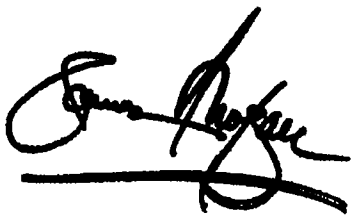
PATENT NO. : 6,405,219 B2
DATED : June 11, 2002
INVENTOR(S) : Christian D. Saether et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 12,
Line 19, "(R1/IDIRMAP.RCA)" should read -- (R1/DIRMAP.RCA) --

Signed and Sealed this

Fifteenth Day of October, 2002



Attest:

Attesting Officer

JAMES E. ROGAN
Director of the United States Patent and Trademark Office

US005978828A

United States Patent [19]

Greer et al.

[11] Patent Number: 5,978,828

[45] Date of Patent: Nov. 2, 1999

[54] **URL BOOKMARK UPDATE NOTIFICATION OF PAGE CONTENT OR LOCATION CHANGES**

[75] Inventors: **Paul E. Greer**, Portland; **Anand Pashupathy**, Beaverton, both of Oreg.

[73] Assignee: **Intel Corporation**, Santa Clara, Calif.

[21] Appl. No.: **08/874,292**

[22] Filed: **Jun. 13, 1997**

[51] **Int. Cl.**[6] **G06F 17/30**; G06F 13/38; G06F 15/167

[52] **U.S. Cl.** **709/2.24**; 709/202; 709/203; 709/206; 709/217; 709/219; 709/224; 709/223; 707/10; 707/501; 707/511; 707/513; 707/530; 707/203

[58] **Field of Search** 707/501, 10, 511, 707/513, 530, 203, 202, 224, 206, 217, 219, 223; 345/329, 330, 331, 334, 333, 335

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,796,393	8/1998	MacNaughton et al.	345/329
5,813,007	9/1998	Nielsen	395/200.36
5,835,712	11/1998	Dufresne	395/200.33

OTHER PUBLICATIONS

Thomas Bell and Fred Douglis, "An Internet Difference Engine and its Applications", IEEE, Compcon 96, 1996.

An Internet Difference Engine and Its Application, Thomas Bell and Fred Douglis., AT&T Bell Lab., Compcon'96 IEEE Computer Society Int'l Conference, Feb. 1996.

"Surfing Corporate Intranets," Zorn et al., Online, v21, May 1997.

Stepping Off the Wire; Surfing Without a Net, PC Week, p. 90, Oct. 1996.

NetAttache Enterprise Server, Enterprise Off-Line, Lenny Bailes, Computerworld, v31, n11, p. 89(2), Mar. 1997.

"Browser Booster", Lenny Bailes, Window Magazine, Mar. 1997.

Net-It Now! 1.5, PR Newswire, Feb. 1997.

Smart Bookmarks 2.0(TM), Aug. 1996.

Internet Mania, Corel Corporation, Nov. 1995.

Internet Access, Tierra Communication, Inc., Aug. 1996.

Tierra Highlights2, Tierra Communications, Inc., Jan. 1997.

New NAPro v.250e, 1996.

NetCarta Webmap, NetCarta Corporation, Nov. 1996.

Netscape Developer's Conference, M2 Presswire, Oct. 1996.

Primary Examiner—Frank J. Asta

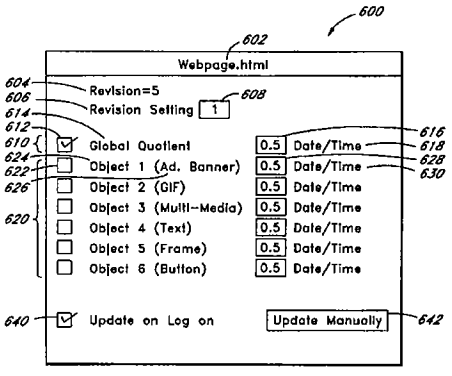
Assistant Examiner—William C. Vaughn, Jr.

Attorney, Agent, or Firm—Blakely, Sokoloff, Taylor & Zafman LLP

[57] **ABSTRACT**

The present invention is an apparatus and method of providing notification of a content change of a web page. The method includes the steps of transmitting a request from a first electronic system to a second electronic system for a quotient value indicative of the content change, transmitting the quotient value from the second electronic system to the first electronic system, comparing the quotient value to a predetermined value to determine whether a threshold is triggered, and notifying the first electronic system of the content change if the threshold is triggered.

20 Claims, 5 Drawing Sheets



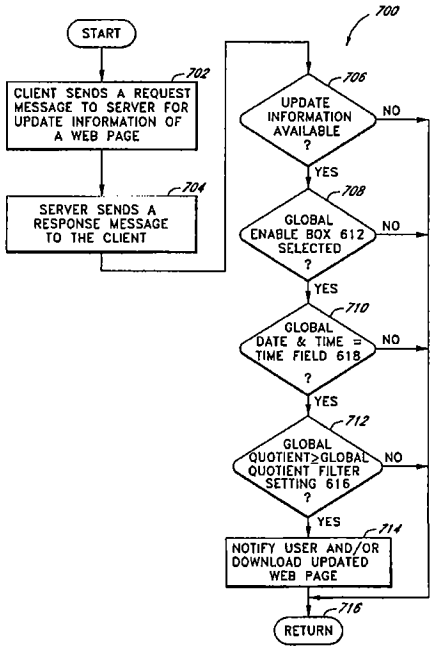


FIG. 1

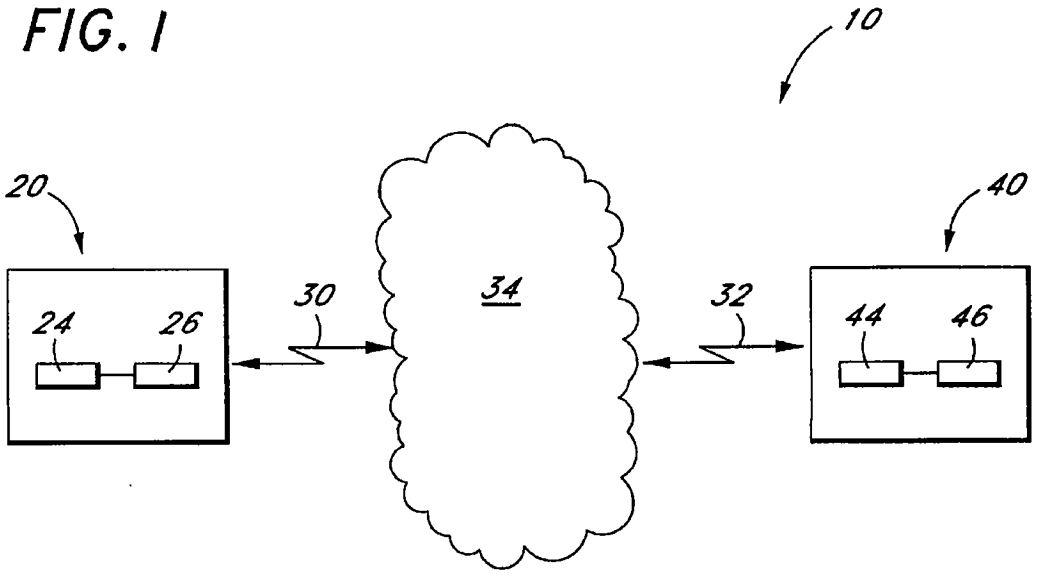


FIG. 2

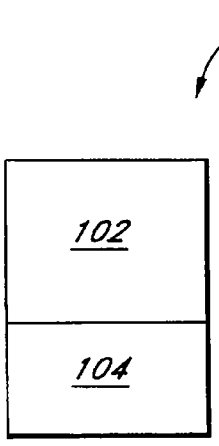


FIG. 3

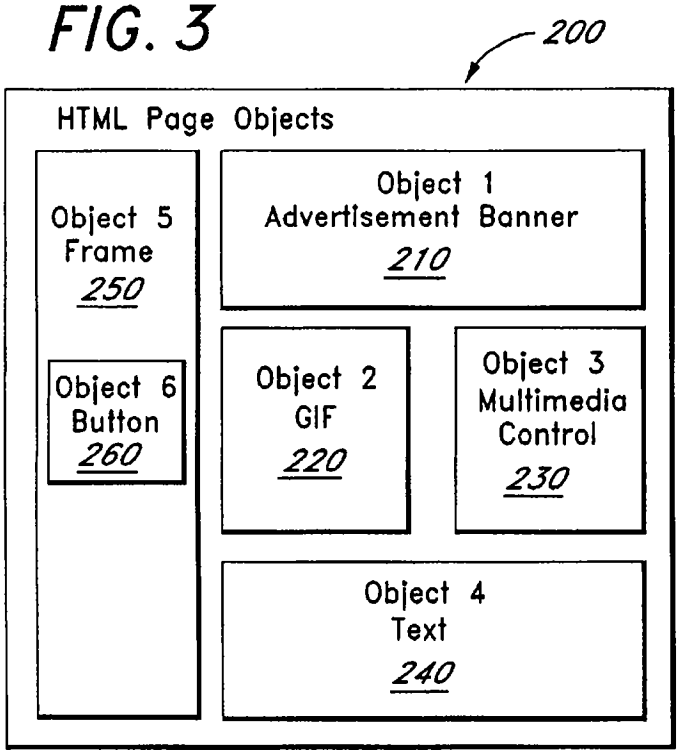

200
HTML Page Objects
Object 5
Frame
250
Object 6
Button
260
Object 1
Advertisement Banner
210
Object 2
GIF
220
Object 3
Multimedia
Control
230
Object 4
Text
240

FIG. 4

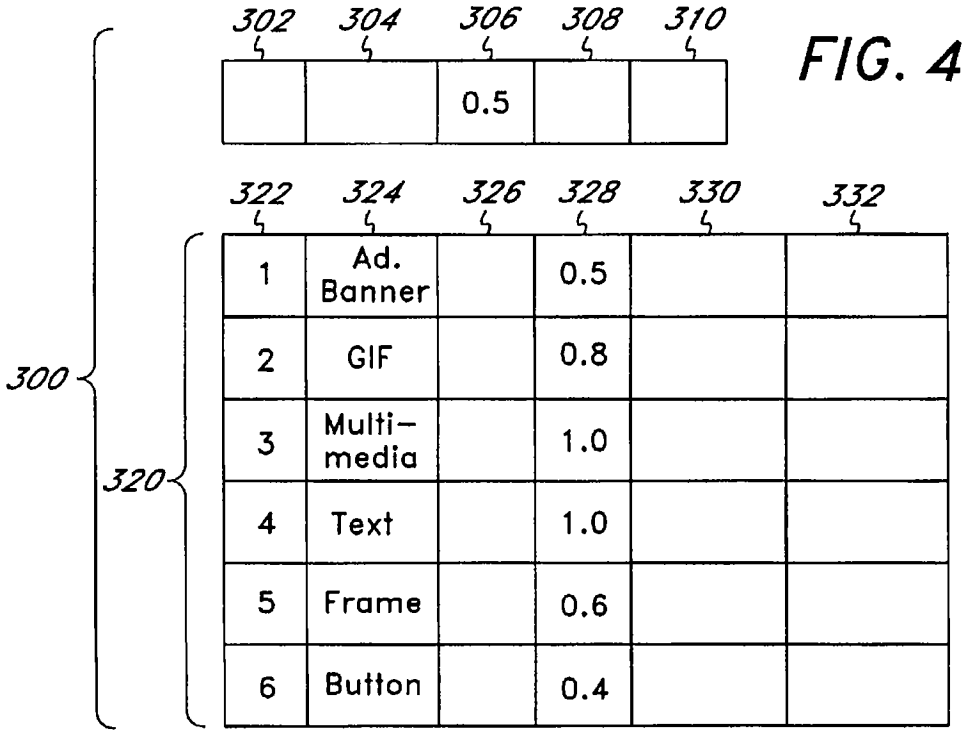

302
304
306
308
310
0.5
322
324
326
328
330
332
300
320
1 Ad. Banner 0.5
2 GIF 0.8
3 Multi-media 1.0
4 Text 1.0
5 Frame 0.6
6 Button 0.4

FIG. 5

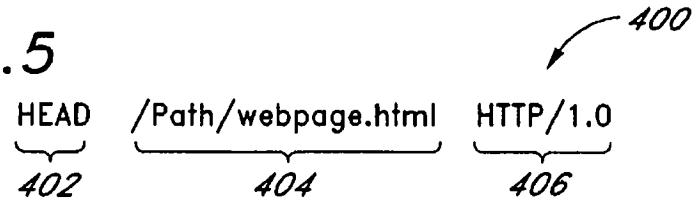


FIG. 6

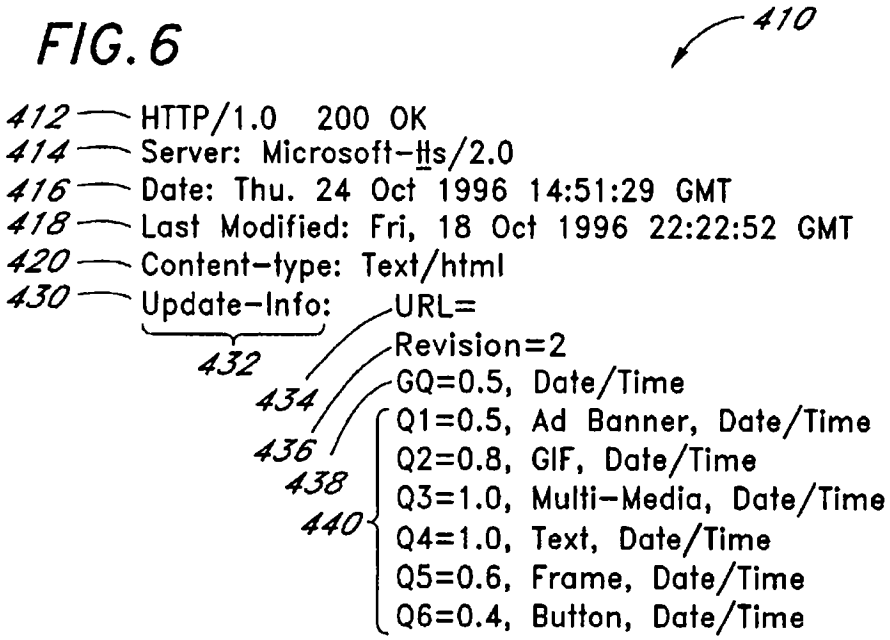


FIG. 7

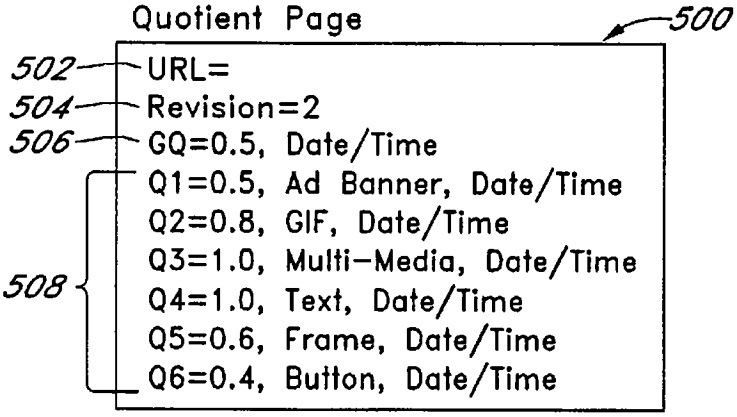


FIG. 8

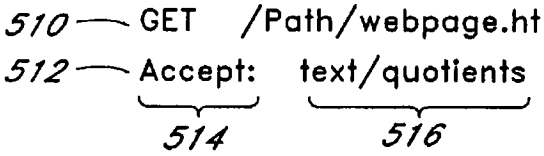
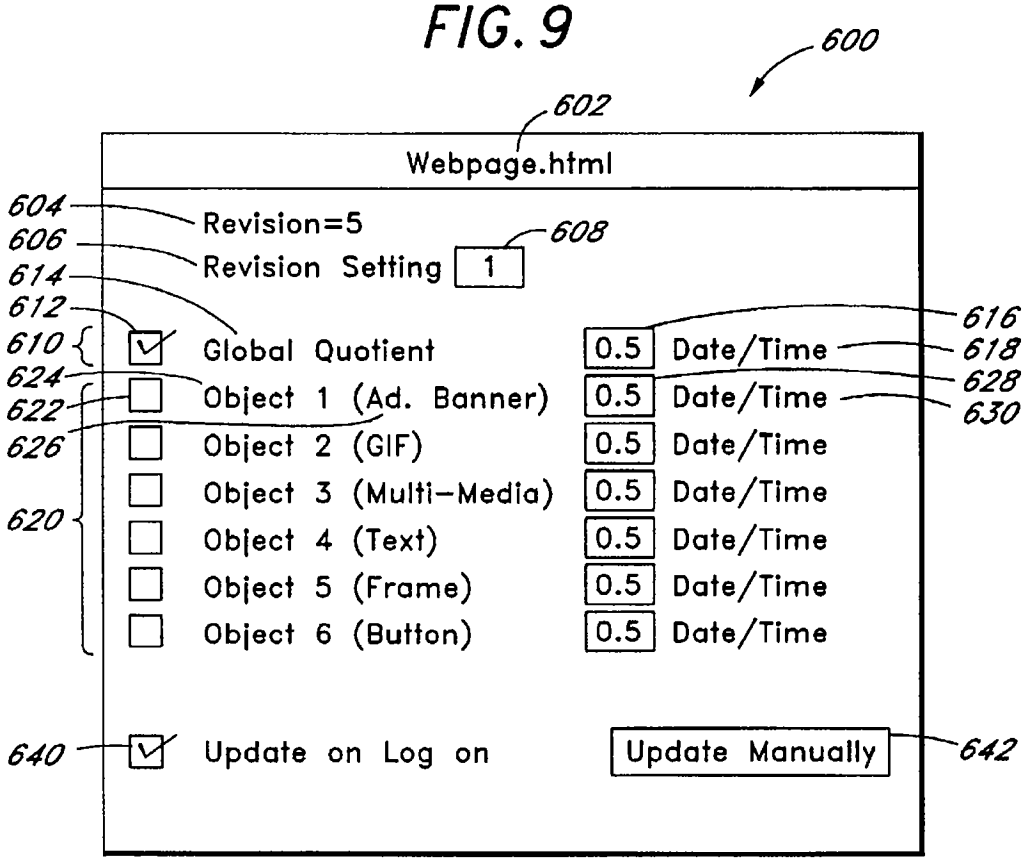


FIG. 9


600
602
Webpage.html
604
Revision=5
606
Revision Setting
608
1
614
612
610
Global Quotient
0.5
Date/Time
616
618
624
622
626
Object 1 (Ad. Banner)
0.5
Date/Time
628
630
Object 2 (GIF)
0.5
Date/Time
620
Object 3 (Multi-Media)
0.5
Date/Time
Object 4 (Text)
0.5
Date/Time
Object 5 (Frame)
0.5
Date/Time
Object 6 (Button)
0.5
Date/Time
640
Update on Log on
Update Manually
642

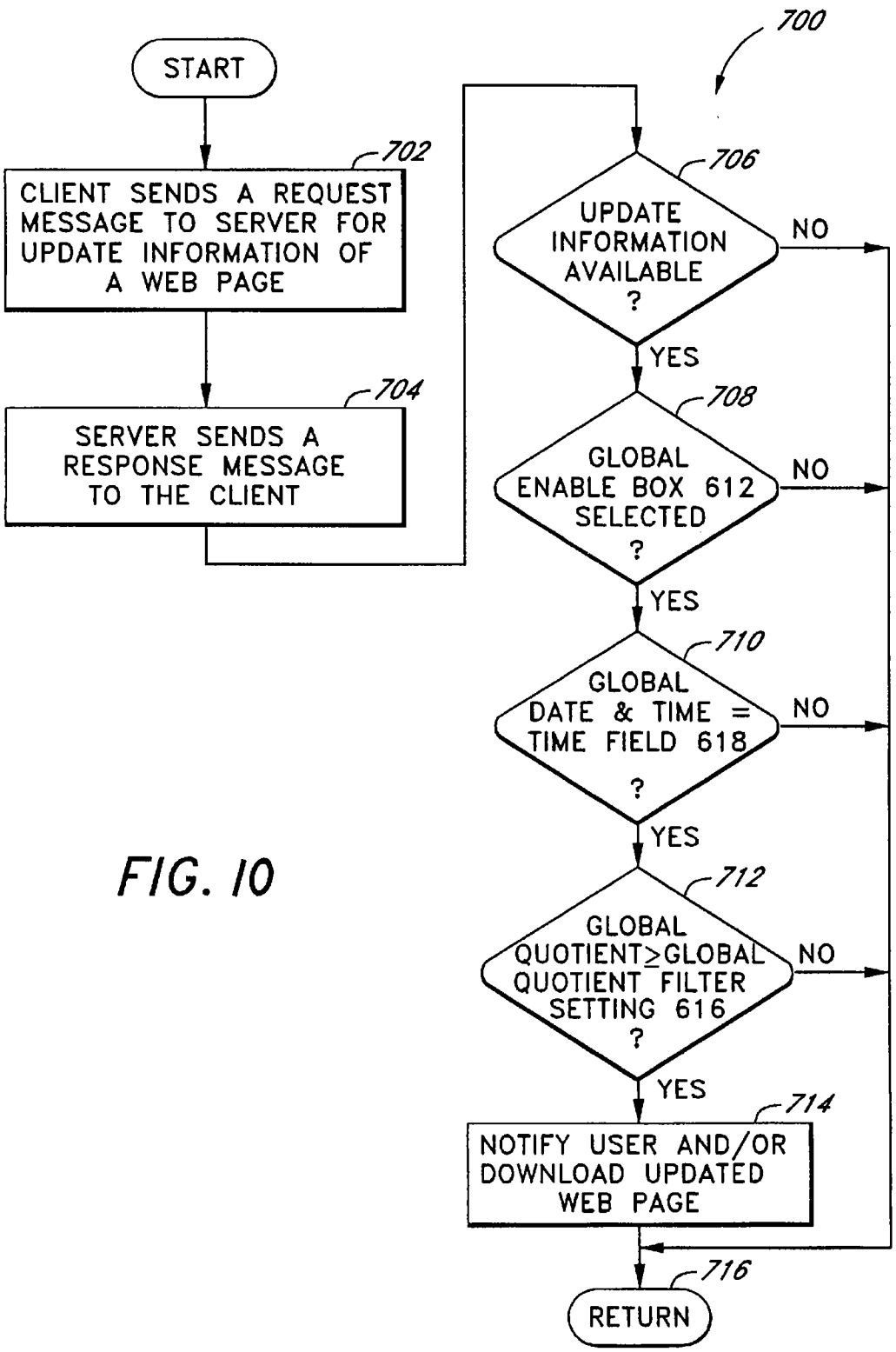

700
START
702
CLIENT SENDS A REQUEST MESSAGE TO SERVER FOR UPDATE INFORMATION OF A WEB PAGE
704
SERVER SENDS A RESPONSE MESSAGE TO THE CLIENT
706
UPDATE INFORMATION AVAILABLE ?
NO
YES
708
GLOBAL ENABLE BOX 612 SELECTED ?
NO
YES
710
GLOBAL DATE & TIME = TIME FIELD 618 ?
NO
YES
712
GLOBAL QUOTIENT≥GLOBAL QUOTIENT FILTER SETTING 616 ?
NO
YES
714
NOTIFY USER AND/OR DOWNLOAD UPDATED WEB PAGE
716
RETURN

FIG. 10

URL BOOKMARK UPDATE NOTIFICATION OF PAGE CONTENT OR LOCATION CHANGES

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to the world wide web. More particularly, the present invention relates to providing update notification of Web page content or location changes.

2. Description of Related Art

The World Wide Web ("Web") is a massive collection of Web pages that are linked together by the Internet, the worlds largest public network. With the Web and the Internet, a user has access to a wealth of diverse and, in many instances, volatile information at his fingertips.

Today, the main piece of software used for connecting to and displaying Web pages on a client is called a Web browser. The main function of a Web browser is to interpret the information received from a Web page and display it on a computer monitor. Most of today's Web browsers have a feature called bookmark (or favorites). This feature allows a user to tag (or save) the address or Uniform Resource Locator ("URL") of a favorite Web page and add a short description of the Web page in an address book. The next time the user wants to connect to his favorite Web page, he selects the Web page from the address book without having to remember or type in the address. Thus, the bookmark feature gives the user a quick way of connecting to his favorite Web page.

However, with Web pages often being constantly updated with new information, there is no feature or mechanism that provides an indication of whether, when, or how much the contents of a Web page has changed. This is important because a user will want the most updated information of a Web page that he has saved or cached. The only way to determine whether a Web page has changed or changed significantly is to download the Web page. This may be a formidable task if the user wants to update several favorite Web pages saved or cached because the user has to download each Web page manually and determine whether the Web page has changed significantly enough to replace the old version.

Therefore, there is a need in the technology for an apparatus and method of providing update notification of Web page content or location changes using existing mechanisms.

SUMMARY OF THE INVENTION

The present invention is an apparatus and method of providing notification of a content change of a web page. The method includes the steps of transmitting a request from a first electronic system to a second electronic system for a quotient value indicative of the content change, transmitting the quotient value from the second electronic system to the first electronic system, comparing the quotient value to a predetermined value to determine whether a threshold is triggered, and notifying the first electronic system of the content change if the threshold is triggered.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

FIG. 1 illustrates one embodiment of an electronic system suitable for use with the present invention.

FIG. 2 illustrates an enhanced browser contained within the client memory element of FIG. 1.

FIG. 3 illustrates a typical Web page contained within the server memory element of FIG. 1.

FIG. 4 illustrates an exemplary change control record contained within the server memory element of FIG. 1.

FIG. 5 illustrates a typical HEAD command request which is transmitted from a client to a server.

FIG. 6 illustrates a typical response message to the HEAD command request transmitted from a server to a client.

FIG. 7 illustrates another embodiment of providing update notification of Web page content or location changes.

FIG. 8 illustrates a request by a client to retrieve the quotient page of FIG. 7.

FIG. 9 illustrates a client filter setup window suitable for use with the present invention.

FIG. 10 is a flow diagram illustrating an exemplary method of obtaining update information of a Web page and notifying a user of the updated information.

DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to an apparatus and method of providing update notification of Web page content or location changes. As discussed herein, an "electronic system" is an apparatus including hardware and/or software for processing data. The electronic system may include, but is not limited to, a computer (e.g., laptop, desktop, server, mainframe, etc.), hard copy equipment (e.g., printer, plotter, fax machine, etc.), banking equipment (e.g., an automated teller machine), and the like. Moreover, a "memory element" is a device that is capable of storing data such as a software program. The memory element may include, but is not limited to, a storage disk (e.g., hard, floppy, compact, optical, etc.), Random Access Memory ("RAM"), Read Only Memory ("ROM"), non-volatile RAM, zip memory, and the like. A "communication link" refers to the channel of communication. The communication link may include, but is not limited to, a telephone line, a satellite connection, an Integrated Services Digital Network ("ISDN") line, an Ethernet connection, a coaxial connection, a fiber optic connection, and the like. Finally, as discussed herein, a Web page or HyperText Markup Language ("HTML") Web page is a data file on a server electronic system (also known as a content site) that contains information.

FIG. 1 illustrates one embodiment of an electronic system **10** suitable for use with the present invention. Referring to FIG. 1, the electronic system **10** includes a client electronic system **20**, a server electronic system **40**, and communication links **30** and **32** coupled to a network **34** (e.g., Internet). The communication links **30** and **32** couple the client electronic system **20** to the server electronic system **40** through the network **34**. The client electronic system **20** includes a client processor **24** coupled to a client memory element **26** and server electronic system **40** includes a server processor **44** coupled to a server memory element **46**. The client and server processors **24** and **44** execute application programs that are stored in the client and server memories **26** and **46**, respectively. As discussed herein, a client electronic system is an electronic system that establishes connections for the purpose of transmitting requests and a server electronic system is an electronic system that accepts connections in order to service requests by transmitting responses. Moreover, a "client" is an application program that establishes connections for the purpose of sending requests and a

"server" is an application program that accepts connections in order to service requests by sending back responses.

FIG. **2** illustrates an enhanced browser **100** contained within the client memory element **26** of FIG. **1**. Referring to FIGS. **1** and **2**, client processor **24** executes the enhanced browser **100** which is contained within the client memory element **26**. The enhanced browser **100** includes a browser **102** (e.g., Microsoft™ Internet Explorer™, Netscape Navigator™, etc.) and a browser enhancement **104**. The browser **102** retrieves and formats Web pages for display on a user's screen or monitor. Implemented in client electronic system **20**, the browser enhancement **104** augments the browser **102** with new capabilities suitable for use with the present invention. Among other things, the browser enhancement **104** includes a setup window having user assignable settings for retrieving update information of a Web page and determining whether the magnitude of change of the Web page warrants alerting the user of the updated information and/or automatically downloading the Web page. The browser enhancement **104** may be a module added to the browser **102**. Alternatively, the browser enhancement **104** may be a stand-alone helper application, a browser "plug-in", a JAVA™ program, or an ActiveX control.

FIG. **3** illustrates a typical organization of a Web page **200** contained within the server memory element **46** of FIG. **1**. Referring to FIG. **3**, Web page **200** includes object **210** (e.g., advertisement banner), object **220** (e.g., graphics image format, "GIF"), object **230** (e.g., multimedia control), object **240** (e.g., text), object **250** (e.g., a frame), and object **260** (e.g., a button). Web page **200** may further include other types of objects such as HTML links, motion images, sound clips, HTML page tables, and other embedded objects (e.g., plug-ins, multimedia, etc.). Each Web page on the server electronic system **46** of FIG. **1** will have a corresponding change control record summarizing the changes of each Web page object. The server memory element **46** of FIG. **1** further includes a Web page change control database. The change control database is a database of all of the change control records.

FIG. **4** illustrates an exemplary change control record **300** contained within the server memory element **46** of FIG. **1**. For sake of clarity, the Web page change control record **300** corresponds to the Web page **200** of FIG. **3**. Referring to FIG. **4**, Web page change control record **300** includes a Web page ID field **302**, a Uniform Resource Locator ("URL") field **304**, a global quotient field **306**, a last update field **308**, and a revision field **310**. The Web page ID field **302** includes a unique identifier specifying that the change control record **300** corresponds to the Web page **200** of FIG. **2**. The URL field **304** specifies whether the address of the Web page has changed and the new address of the Web page. The global quotient field **306** includes a global quotient value which specifies the magnitude of change of the overall Web page since the last update. The global quotient value ranges from 0.0, meaning no change, to 1.0, meaning a 100% change. The last update field **308** specifies the date and time of the most recent update of the Web page.

The revision field **310** includes a value that specifies the revision number of the Web page. When a Web page is created, the value in the revision field **310** will typically be **0**. For every update of the Web page, this value will be changed to reflect the update (i.e., by some predetermined protocol). For example, the value in the revision field **310** will be changed to 5 after the fifth revision. There may be other fields defined in the Web page change control record **300** that provide more information regarding the updates of the Web page. By way of example, the Web page change control record **300** may include one or more other fields that specify the magnitude, revision number, and date and time of last modification of one or more previous major updates. The one or more fields will inform the user of intermediate major updates between the last update the user downloaded and the current update.

Continuing to refer to FIG. **4**, Web page change control record **300** further includes one or more object fields **320** corresponding to the number of Web page objects. For instance, the Web page change control record **300** for the Web page **200** of FIG. **3** includes six object fields corresponding to each of the objects **210**, **220**, **230**, **240**, **250**, and **260**. In particular, each object field **320** includes an object number field **322**, an object description field **324**, a change control algorithm field **326**, an object quotient field **328**, a last update field **330**, and unspecified fields **332**. The object number field **322** specifies the name or number of the Web page object (e.g., 1, 2, etc.). The object description field **324** describes the Web page object such as the type (e.g., GIF) and size of the object, the nature of the change, and the like. The change control algorithm field **326** specifies the change control algorithm assigned to the object. The object quotient field **328** includes an object quotient value which specifies the magnitude of change of the particular object since the last update. This value is interpreted by a client filter contained within the client electronic system **20** of FIG. **1** (described below). The object quotient is a weighted average derived from the change control algorithm. The object quotient is always between 0.0 (no change) to 1.0 (100% change). The last update field **330** specifies the most recent update of that particular object. Finally, the unspecified fields **332** allows for expansion of new fields in the future.

In one embodiment, the change control algorithm in the change control algorithm field **330** is used to determine the object quotient value. To assign an object quotient value, an algorithm is used that matches the nature of the change. For example, an algorithm which deals with text will be different than one that deals with an embedded media object. This is because not all bytes are equal in value. This is also to insure that the algorithms chosen bring all the changes to a common ground and in tune with the object quotient values. The nature of the change is managed or represented by the algorithm chosen. For example, a text based algorithm could be assigned to calculate the key word associations weighted against the context of the overall idea in whose context the words are being used. An embedded image will have a total different algorithm than text. In one embodiment, there is a base line created for each page that will be the basis of comparison. The base line may be updated as the Web page is updated. As an illustration, each object type on a page is assigned a keyword or a description. The system will compare keywords and decide how much it has changed from the base line. The system has semantic knowledge associated with each article or each object in a page.

As part of the creation and updating of a Web page, the Web page author (or system administrator) may use server software tools (e.g., a script) to build and/or edit the Web page change control database **300**. That is, the database creation/editing is associated with the creation and editing of the Web page. For example, using a change control algorithm, the Web page author assigns a global quotient value and a plurality of object quotient values and the categorical nature of each change for interpretation by a client filter running on the client electronic system **20** of FIG. **1**. A server electronic system transmits this information to a client electronic system using a number of different methods as will be described below. In one embodiment, the

quotient values are embedded into a HyperText Transfer Protocol ("HTTP") header of a Web page.

FIG. **5** illustrates a typical HEAD command request which is transmitted from a client electronic system to a server electronic system. The HEAD command (or method) request is defined in the HTTP/1.0 (and HTTP/1.1) specification (request for comment, "RFC" 1945) and specifies to a server to only return the header information for the indicated Web page. Referring to FIG. **4**, the HEAD method request field **400** includes three sub-fields. The first sub-field **402** is the command HEAD which specifies to the server to retrieve only the header of a Web page and not the body. The second sub-field **404** specifies the location of the Web page. Finally, the third sub-field **406** specifies the protocol and version currently running on the client.

FIG. **6** illustrates a typical response message to the HEAD command request transmitted from a server electronic system to a client electronic system. Referring to FIG. **6**, the response message **410** to the HEAD command request includes status field **412** which specifies the protocol and version that is supported, server field **414** which specifies the type and version of the server, time stamp field **416** which indicates the access date and time, last update field **418** which specifies the data and time of last modification of the Web page, and content-type field **420**. The content-type field **420** specifies the media type (type/subtype) of the body of the Web page that is to follow (e.g., text/html). Media types are discussed in Multipurpose Internet Mail Extensions ("MIME"), as defined by the Internet Engineering Task Force Document, RFC 1521. Most of the information in the response message **410** is optional with the exception of the content-type field.

In addition, the response message **410** of the present invention includes a quotient field **430** which includes an identifier **432**, an optional URL field **434**, a revision field **436**, a global quotient field **438**, and one or more object quotient fields **440**. In one embodiment, the information included in the quotient field **430** is obtained from a change control record contained within the server electronic system. The identifier **432** (e.g., Update-Info) identifies that web page update information is to follow. The optional URL field **434** is included in the quotient field **430** when the URL of the Web page has changed. The URL field **434** includes an identifier ("URL") followed by a new address of the Web page. If the address of the Web page has not changes, then typically nothing will follow the identifier. In such a case, this field may be excluded. The revision field **436** includes an identifier (e.g., "revision") followed by a revision value of the Web page which corresponds to the value in the revision field **310** of FIG. **4**. The global quotient field **438** includes an identifier ("GQ") followed by a global quotient value and a date and time of last modification corresponding to fields **306** and **308** of FIG. **4**, respectively.

Continuing to refer to FIG. **6**, the one or more object quotient fields **440** include one or more object identifiers ("Qx") corresponding to the one or more objects in the Web page **200** of FIG. **3**. Each object quotient identifier is followed by an object quotient value, an object type, and the date and time of last modification for that particular object. These values correspond to fields **328**, **324**, and **330** of FIG. **4** respectively. The global quotient value and all object quotient values, vary between 0.0 to 1.0 and each is set by the Web page creator/administrator. More specifically, the global quotient value indicates the magnitude of change for the entire Web page and each object quotient value indicates the magnitude of change for the corresponding object. The quotient field **430** can be added, inserted, or embedded into the response header **410**. This process is performed either manually or by an automated process on the server electronic system **40** of FIG. **1**. The quotient field **430** can also be generated and appended to the response header **410** on the fly (i.e., when a server receives a request for the header) by a server script.

FIG. **7** illustrates another embodiment of providing update notification of Web page content or location changes. In this embodiment, the quotient field **430** of FIG. **6** is added or inserted in a quotient page **500** rather than in the HTTP header of the Web page. The quotient page **500** and its contents are contained within the server memory element **46** of the server electronic system **40** as shown in FIG. **1**. Referring to FIG. **7**, the quotient page **500** includes an optional URL field **502** which includes a new address of the Web page when the URL of the Web page has changed, a revision field **504** which includes a value that specifies the revision of the Web page, and a global quotient field **506** which includes an identifier (e.g., "GQ") followed by a quotient value and a date and time of last modification. In addition, the quotient page **500** includes one or more object quotient fields **508**. Each object quotient field includes an object identifier (e.g., "Qx") followed by the object quotient value, an object type, and a date and time of last modification for that particular object.

The quotient page **500** can be created/updated when the Web page is created/updated by the use of a script. When the Web page and Web page change control record are updated, a script can be used to update the quotient page **500** automatically. The script retrieves the information from the Web page change control record and inserts the information in the quotient page **500** as shown, for example, in FIG. **7**. Alternatively, the quotient page **500** may be updated on-demand. That is, the information is only updated as requested by a client. It is contemplated that the quotient page **500** may contain other information.

The quotient page **500** is assigned a MIME type format so that a client can specifically request the quotient page **500** by the MIME type format. MIME describes the contents of a document by referring to a standardized list of document types organized by type and subtype. As part of the present invention, an exemplary new MIME type called "text/quotients" is created to identify and distinguish the quotient page **500** from the Web page **200**. Although existing browsers may not be able to handle this new MIME type file format, the browsers typically launch helper applications (or browser plug-ins) that can handle the new file format. The browser enhancement **104** of FIG. **2** is such a helper application or browser plug-in.

FIG. **8** illustrates a request by a client to retrieve the quotient page **500** of FIG. **7**. FIG. **8** includes a GET command request field **510** and a request header field **512**. The GET command request specifies to a server to retrieve whatever information is identified by the request header field **512**. The request header field **512** includes an identifier **514** followed by a type/subtype field **516**. The identifier ("Accept") **514** indicates to the server the document types the client wants to receive. The type/subtype field **516** indicates to the server that the client wants to receive "text/quotients" document types. When the server receives the request field **510** and request header field **512**, the server determines that the only acceptable media type is "text/quotients" and transmits the quotient page **500** of FIG. **7** rather than the Web page **200**.

FIG. **9** illustrates a client filter setup window suitable for use with the present invention. Referring to FIG. **9**, the

software used to generate the setup window **600** is contained within and executed by the client memory element **26** and the client processor **24** respectively of FIG. **1**. In one embodiment, the application program used to generate the setup window **600** is the browser enhancement **104** of FIG. **2**. The setup window **600** includes a name field **602** which includes a URL or a description of a Web page. For sake of clarity, the setup window **600** will be described with respect to the Web page **200** of FIG. **3**. The setup window further includes a global field **610** which includes the global quotient setting, and an object field **620** which includes the settings for each Web page object. In particular, the global field **610** includes a global enable box **612**, an identifier **614** (e.g., "Global Quotient" or "Global Change Filter"), a global quotient filter setting **616** which specifies the global quotient filter value, and a global time field **618** which specifies the most recent date and time of modification of the Web page. The enable box **612**, when selected, specifies to the client to compare the global quotient value with the filter value when update information is downloaded to the client.

The setup window **600** includes a revision field **604** which specifies the last revision of the Web page that was downloaded. The setup window **600** further includes field **606** having a user changeable value as shown in box **608**. If the current revision minus the last revision (value in field **604**) is greater than or equal to the value shown in box **608**, then the Web page is automatically download. This gives the user flexibility in automatically downloading the Web page after the Web page has gone through a predetermined number of revisions. There may be other user changeable settings in the setup window **600** that allow the Web page to be automatically downloaded. For example, this may occur if an intermediate update has a large enough global quotient value (e.g., greater than the global quotient filter setting **616**).

Still referring to FIG. **9**, the object field **620** includes a plurality of object sub-fields corresponding to the number of Web page objects. Each object sub-field includes an object enable box **622**, an object identifier **624** (e.g., "Object 1"), an object type field **626** which specifies the object type (e.g., text), an object quotient filter setting **628** which specifies the Web page object quotient filter value, and a time sub-field **630** which specifies the date and time of last modification of the Web page object. In addition, the setup window **600** includes an auto update log-on field **640** and a manual update field **642**. If the auto update log-on field **640** is selected, the client obtains the updated information of the Web page every time a user logs on to the Internet. Two exemplary mechanisms for obtaining the update information of the Web page are shown and described in the description accompanying FIGS. **5–8**. The manual update field **642** allows the user to manually invoke the client to obtain the update information of the Web page.

Once a Web page has been downloaded and stored or cached on a client electronic system, a user may tag the Web page for obtaining update information. This may be done by, for example, selecting a menu item in the client (e.g., enhanced browser **100** of FIG. **2**) called "updates". Once the menu item is selected, a window such as the setup window **600** of FIG. **9** is displayed. The user can select the global enable box **612** and/or one or more of the object enable boxes **622** in order to request the client to obtain update information of the Web page. However, if the global enable box **612** and all of the object enable boxes are not selected, the client will not request for update information of the Web page. The user may request the client to obtain update information on log on (e.g., by selecting the enable box **640**), manually (e.g., by selecting the manual update field **642**), or periodically (e.g., every fifth log-on, beginning of the month, etc.).

Initially, the setup window **600** loads default values (e.g., 0.5) for the global quotient filter setting **616** and each of the object quotient filter settings **628**. Moreover, initially, the object identifier **624**, the object type **626**, and the global and object time fields **618** and **630** are not known and will be updated once the client obtains the update information for the first time. The setup window **600** allows user assignment of filter values to determine what magnitude and categorical nature of the change will trigger a change alert to the user. The user can select a different threshold value for the global quotient filter setting **616** and each of the object quotient filter settings **628**. For example, the user may only enable the global quotient filter setting **616** by selecting the global enable box **612**. Alternatively, the user may only want to be notified when there is a 60% change to the text of the Web page **200** of FIG. **3**. This is done by only selecting the object enable box **622** for object **4** and changing the filter value in the object quotient filter setting **628** to 0.6. The client can be set to poll one or more Web pages that have been tagged or cached. Moreover, the client filter values can be automatically tuned and adapted using intelligent software that tracks user preference traits by observing their Web browsing patterns.

The client filter will also scan the optional URL field, if any, to determine whether the address of the Web page has changed. If the address of the Web page has changed, then the client filter will alert the user of that fact, for example, by displaying a pop-up window. The user will be prompted and may choose to update the bookmarked/tagged URL with the updated URL and/or download the Web page.

FIG. **10** is a flow diagram illustrating an exemplary method of obtaining update information of a Web page and notifying a user of the updated information. Referring to the flow diagram **700**, the process begins a Step **702** where a client sends a request message to a server for update information, if any, of a Web page located on the server. At Step **704**, the server sends a response message to the client. At Step **706**, if the response message includes update information, then the process continues to Step **708**, otherwise the process is ended (Step **716**). This may occur if a server does not include update information on Web pages contained within the server memory. Once the update information is obtained, the process determines whether the global enable box **612** of FIG. **9** is selected by a user (Step **708**). If the global enable box **612** (or any of the object enable boxes **622**) is/are selected, then the process proceeds to Step **710**, otherwise the process is ended (Step **716**). Assuming that only the global enable box **612** is selected, at Step **710**, the process determines whether the global date and time of last modification included in the update information is equal to the global time field **618**. If not, the process continues to Step **712**, otherwise the process is ended (Step **716**) meaning that the Web page has not been modified since the last time it was checked.

Continuing to refer to FIG. **10**, at Step **712**, if the global quotient value included in the update information is less than (e.g., or less than or equal to) the global quotient filter setting **616**, then the process is ended (Step **716**). This means that the magnitude of change was not high enough to warrant alerting the user and/or downloading the Web page. On the other hand, if the global quotient value is greater than or equal to (or just greater than) the global quotient filter setting **616**, the process proceeds to Step **714**. At Step **714**, the process notifies the user by displaying a pop-up window. The process can also be configured to simply download the Web page in addition to or in lieu of notifying the user. Additionally, for each object enable box **622** that is selected, the process Steps **708–714** are performed for that particular object.

When notified of the updated information, a user, using the client (e.g., enhanced browser **102** of FIG. **2**), can query the nature of the Web page changes, such as a change to the entire Web page or one or more objects in the Web page. In addition, the user can query the magnitude of the change, the categorical nature of the change (e.g., based on the user setting, the update is urgent, interesting, FYI), and the date and time of the change. The user can then update his PC with the updated Web page (i.e., update the old Web page cached or stored on a local memory with the new Web page).

While certain exemplary embodiments have been described and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative of and not restrictive on the broad invention, and that this invention not be limited to the specific constructions and arrangements shown and described, since various other modifications may occur to those ordinarily skilled in the art.

What is claimed is:

1. A method of indicating a content change of a web page, comprising:

transmitting a request from a first electronic system to a second electronic system for a quotient value indicative of the content change of the web;

responsive to receiving the request, transmitting said quotient value from said second electronic system to said first electronic system;

said first electronic system comparing said quotient value to a predetermined value to determine whether a threshold is triggered; and

transmitting a second request from said first electronic system to said second electronic system for the Web page if the threshold is triggered.

2. The method of claim **1** further comprising notifying said first electronic system of the content change if said threshold is triggered.

3. The method of claim 1 further comprising transmitting the web page from said second electronic system to said first electronic system.

4. The method of claim **1** wherein prior to transmitting said quotient value from said second electronic system to said first electronic system, the method comprises adding said quotient value to a HTTP header of the web page on said second electronic system.

5. The method of claim **4** wherein transmitting said request comprises transmitting a HEAD command request from said first electronic system to said second electronic system and wherein transmitting said quotient value comprises transmitting said HTTP header including said quotient value from said second electronic system to said first electronic system.

6. The method of claim **1** wherein prior to transmitting said quotient value, the method comprises adding said quotient value to a file on said second electronic system.

7. The method of claim **6** wherein transmitting said request comprises transmitting a GET command request from said first electronic system to said second electronic system and wherein transmitting said quotient value comprises transmitting said file including said quotient value from said second electronic system to said first electronic system.

8. The method of claim **1** wherein said quotient value ranges between 0.0 to 1.0 and said predetermined value is set between 0.0 to 1.0.

9. The method of claim **1** wherein comparing said quotient value to said predetermined value comprises determining whether said quotient value is greater than or equal to said predetermined value.

10. An electronic system that indicates a content change of a web page, comprising:

a communication link;

a first electronic system coupled to said communication link, said first electronic system includes a memory element having a file and a web page, said file includes a quotient value indicative of a magnitude of change of the web page, said first electronic system to transmit said file in response to a request, and to transmit said web page in response to a second request; and

a second electronic system coupled to said communication link, said second electronic system to transmit said request to said first electronic system to retrieve said quotient value in said file, said second electronic system to transmit said second request to said first electronic system to retrieve said web page if said quotient value is greater than or equal to a predetermined value.

11. The electronic system of claim **10** wherein said file is a HTTP header of the web page.

12. The electronic system of claim **11** wherein said request is a HEAD command request.

13. The electronic system of claim **10** wherein said file is a quotient page of the web page.

14. The electronic system of claim **13** wherein said request is a GET command request.

15. A method of notifying a content change of a web page, having one or more objects, comprising:

transmitting a request from a first electronic system to a second electronic system for a file containing a global quotient and one or more object quotients, the global quotient and the one of more object quotients being indicative of a magnitude of change of the web page and the one or more objects, respectively;

responsive to receiving said request, transmitting the file from the second electronic system to the first electronic system;

said first electronic system comparing the global quotient to a predetermined global quotient value and the one or more object quotients to a corresponding one or more predetermined object quotient values to determine whether a threshold is triggered; and

transmitting a second request from said first electronic system to said second electronic system for the web page if the threshold is triggered.

16. The method of claim **15** wherein prior to transmitting the file, the method comprises adding the global quotient and the one ore more object quotients to a HTTP header of the web page.

17. The method of claim **16** wherein transmitting said request comprises transmitting a HEAD command request from the first electronic system to the second electronic system and wherein transmitting the file comprises transmitting the HTTP header including the global quotient and the one or more object quotients from the second electronic system to the first electronic system.

18. The method of claim **15** wherein prior to transmitting the file, the method comprises adding the global quotient and the one or more object quotients to the file on the second electronic system.

19. The method of claim **18** wherein transmitting the request comprises transmitting a GET command request from the first electronic system to the second electronic system and wherein transmitting the file comprises transmitting the file including the global quotient and the one or more object quotients from the second electronic system to the first electronic system.

20. The method of claim 15 wherein comparing the global quotient to the predetermined global quotient value and the one or more object quotients to the corresponding one or more predetermined object quotient values to determine whether the threshold is triggered comprises determining whether the global quotient is greater than or equal to the predetermined global quotient value, and determine whether the one or more object quotients are greater than or equal to the corresponding one or more predetermined object quotient values, and wherein transmitting the second request comprises transmitting the second request from said first electronic system to said second electronic system if the global quotient is greater than or equal to the predetermined global quotient value or at least one of the one or more object quotients is greater than or equal to at least one of the corresponding one or more predetermined object quotient values.

* * * * *

IX. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.